Alma Mater Studiorum - Università di Bologna
Campus di Cesena

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

TRAIN YOUR MIND - SOFTWARE PER IL TRAINING COGNITIVO NELL'INVECCHIAMENTO SANO E PATOLOGICO

Relazione finale in
PARADIGMI DI PROGRAMMAZIONE E SVILUPPO

Relatore
Prof. MIRKO VIROLI
Correlatore
Prof. ALESSIO AVENANTI

Presentata da
ELISA CASADIO

Prima Sessione di Laurea
Anno Accademico 2018-2019

Ai miei genitori
Rita e Davide

Sommario

Lo scopo di questa tesi è la realizzazione di un software applicativo per il training cognitivo di soggetti anziani sani o affetti da patologie. L'introduzione di questo strumento di allenamento cognitivo computerizzato vuole permettere al medico e al paziente di svolgere le sessioni di allenamento in modo semplice, focalizzandosi su specifiche aree cognitive. Il progetto è corredato da un framework degli esercizi, che permette allo sviluppatore di creare i propri esercizi e di aggiungerli per poterli eseguire durante le sessioni di allenamento. Per lo sviluppo del sistema, sono state adottate tecnologie note e standard, come Java, che ne permettono una facile estensione e modifica da parte degli sviluppatori. Il lavoro di sviluppo è stato corredato da una sperimentazione sul campo, che ha permesso di verificare se, tramite il software, era possibile distinguere i soggetti sani da quelli malati. Nel testo sono analizzati i requisiti che il sistema deve soddisfare, le scelte progettuali e quelle implementative, gli strumenti di sviluppo adottati e la sperimentazione svolta su un campione di soggetti. Il risultato ottenuto è un sistema che fornisce tutte le funzionalità di base per lo svolgimento delle sessioni di training. Inoltre, l'interfaccia realizzata, il cui aspetto si ispira al mondo del web, consente all'utente finale di utilizzare il software in modo semplice. All'interno del sistema è già presente un set di esercizi da poter svolgere, che è stato commissionato dai ricercatori del Centro Studi e Ricerche in Neuroscienze Cognitive di Cesena.

Indice

Introduzione

Il progetto "Train your mind" consiste nella realizzazione di un software applicativo che consenta di svolgere allenamento cognitivo su soggetti anziani sani o affetti da patologie.

Il crescente invecchiamento della popolazione ha evidenziato il sorgere di nuove problematiche nei soggetti anziani, con un graduale declino delle capacità cognitive. Il progredire di questo declino può portare al verificarsi di patologie ben più gravi, che al giorno d'oggi non hanno ancora una cura.

Il training cognitivo è un intervento neuropsicologico utilizzato in psicologia, neurologia e medicina riabilitativa con lo scopo primario di ritardare le prime fasi del declino cognitivo associato alle patologie neurodegenerative. Inoltre, è in grado di migliorare anche le prestazioni cognitive in soggetti anziani sani. L'allenamento consiste nell'assegnare al paziente lo svolgimento di un compito, permettendogli di addestrare una specifica area cognitiva. Negli ultimi anni, questa tipologia di allenamento è stata oggetto di diversi studi, i quali hanno confermato la sua efficacia, anche in soggetti affetti da decadimento cognitivo lieve (MCI).

Con la necessità di allenare più domini cognitivi contemporaneamente, ampliare le aree cognitive da allenare e avere una completa personalizzazione della sessione di training, sono nati nuovi approcci. Uno fra questi è il training cognitivo computerizzato, il quale sfrutta le moderne tecnologie per erogare i compiti da assegnare ai pazienti. Il paziente potrà allenare le abilità al massimo delle sue capacità divertendosi, grazie allo svolgimento di semplici "giochi". Purtroppo le potenzialità offerte da questi strumenti non sono ancora state sfruttate a pieno, ma si ritiene che possano portare a molti benefici, soprattutto in presenza di allenamenti duraturi nel tempo.

I software di training cognitivo esistenti sul mercato, spesso, non permettono di allenare in modo efficace le funzioni esecutive dei soggetti. Per questo motivo, è nata l'esigenza di realizzare un nuovo software che avesse come elemento di novità la fruizione di esercizi specifici per l'allenamento di certe aree cognitive, ma che permettesse anche di essere esteso con nuove funzionalità.

L'obiettivo del progetto è stato quello di ottenere un software, con un'interfaccia grafica user-friendly, che permettesse la fruizione in modo semplice e veloce delle sessioni di allenamento ai pazienti. Gli esercizi devono essere organizzati all'interno di un framework, in modo da permettere a chiunque ne abbia la necessità di aggiungerne di nuovi rispetto a quelli già implementati di base. Per lo sviluppo del progetto sono, quindi, stati adottati strumenti e tecnologie note e standard, per rendere il sistema facilmente estensibile e modificabile.

Al termine della fase realizzativa del progetto, è stata svolta una sperimentazione sul campo del software. La sperimentazione ha consentito di verificare l'effettiva usabilità del sistema e di ricevere feedback. Inoltre, si è voluto verificare che i dati raccolti dal software permettessero di fare una distinzione netta tra i soggetti sani e quelli malati.

L'elaborato di tesi è stato suddiviso in sei capitoli, che permettono di ripercorrere tutte le fasi di realizzazione e sperimentazione svolte. A questi capitoli, seguono le conclusioni e gli sviluppi futuri.

- Il primo capitolo fornisce una panoramica sul training cognitivo, andando ad analizzare alcune patologie su cui si può applicare e gli strumenti utilizzati.

- Il secondo capitolo analizza i requisiti del sistema richiesti dal committente, definendone il comportamento e le qualità che deve possedere.

- Il terzo capitolo descrive la fase di progettazione dell'architettura e dell'interfaccia grafica, approfondendo le motivazioni che hanno portato a tali scelte.

- Il quarto capitolo illustra tutti gli strumenti e i metodi di sviluppo adottati durante il corso della realizzazione del progetto.

- Il quinto capitolo si sofferma sulla fase implementativa del sistema, andando ad approfondire le scelte implementative che hanno portato all'ottenimento del software.

- Il sesto capitolo tratta la sperimentazione del software, analizzando il campione di soggetti, le metodologie utilizzate e i risultati ottenuti.

Capitolo 1

Training cognitivo

Il cervello è considerato l'organo più complesso e misterioso dell'organismo, in quanto è in grado di controllare molte funzioni, tra loro molto diverse, quali la memoria, il movimento, il linguaggio, la respirazione e il battito cardiaco.

Fino a non molto tempo fa, si credeva che gli esseri umani fossero in grado di utilizzare solamente le funzionalità che il cervello aveva acquisito alla nascita. A seguito di studi approfonditi, è diventato evidente che il cervello è in grado di adattarsi e sviluppare nuove abilità durante tutto il corso della vita di una persona. Questa sua capacità di creare nuovi percorsi e connessioni viene meno con il progredire dell'età o quando si presentano patologie che provocano la degenerazione progressiva.

Secondo uno studio condotto negli Stati Uniti, entro 20 anni, gli anziani costituiranno circa il 25% della popolazione. Il crescente invecchiamento della popolazione evidenzia la necessità di individuare soluzioni che permettano di ritardare il declino cognitivo patologico associato all'invecchiamento.

Il *training cognitivo* è, quindi, uno strumento utilizzato da psicologi, neuropsicologi o psichiatri come medicina riabilitativa. Gli esercizi svolti permettono di raggiungere obiettivi terapeutici, che portano a un possibile miglioramento nelle abilità cognitive del paziente, come ad esempio l'attenzione, la memoria di lavoro, la capacità di risoluzione dei problemi, ... Con questi interventi di training cognitivo è possibile vedere un miglioramento negli anziani sani, che si protrae fino a 5 anni dopo l'allenamento.

L'allenamento cognitivo su anziani sani, infatti, permette di aumentare l'attività dell'ippocampo (parte del cervello responsabile della formazione delle memorie esplicite, della trasformazione da memoria a breve termine in memoria a lungo termine e della navigazione spaziale) e del lobo frontale. È, infatti, l'atrofia all'ippocampo uno dei problemi che si verificano in presenza del morbo di Alzheimer.

I programmi di allenamento tradizionali sono solitamente forniti in modo individuale o di gruppo da istruttori esperti, che decidono le abilità da addestrare, la lunghezza e la frequenza dell'allenamento e le strategie da applicare. In molti casi è necessario un incontro faccia a faccia, perciò si devono avere a disposizione delle sedi in cui incontrarsi e bisogna affrontare un viaggio per spostarsi. Questa modalità può quindi essere molto costosa, sia in termini di personale, che di attrezzature e materiali.

Un'alternativa a questo approccio, può essere quella di utilizzare il computer per svolgere l'allenamento, perché permette di personalizzare gli esercizi da svolgere, fruire una formazione multidominio e monitorare i progressi in tempo reale. In questo modo, il paziente può sviluppare le abilità cognitive al massimo delle sue capacità, grazie allo svolgimento di "giochi" che può ritenere anche divertenti.

1.1 Mild Cognitive Impairment (MCI)

L'avanzamento degli anni, spesso, è automaticamente sinonimo di perdita delle capacità cognitive e viene dato per scontato che la demenza sia l'ovvia conseguenza dell'essere anziani. In realtà, un invecchiamento sano, per quanto porti a un graduale declino delle facoltà mentali, prevede un mantenimento delle principali funzioni cognitive, come ad esempio la memoria, l'attenzione e il linguaggio. La popolazione, però, non può essere divisa tra anziani sani e anziani affetti da malattie neurodegenerative.

Il *decadimento cognitivo lieve* o MCI [11] è una sindrome che provoca un declino cognitivo maggiore del previsto rispetto all'età dell'individuo, ma che non va ad interferire sulle sue normali e quotidiane attività.

Il MCI è da considerarsi un sintomo che, nella maggior parte dei casi, precede la malattia di Alzheimer o altri tipi di demenza. Questa sindrome si distingue dalla demenza, perché i deficit riscontrati sono meno gravi e diffusi e non hanno un riscontro diretto sulla vita quotidiana.

Molti studiosi hanno provato a trovare una definizione per descrivere cosa fosse il decadimento cognitivo lieve. Nel XIX secolo, Prichard fu il primo ad identificarlo e lo definì come il primo stadio di demenza, in cui si verifica una menomazione della memoria più recente, mentre rimane intatta la memoria remota. Un secolo dopo, Kral lo definì da un punto di vista completamente diverso. Secondo lui, i dati e le esperienze, che i soggetti affetti da MCI dimenticavano, appartenevano principalmente al loro passato remoto, invece che al recente passato.

I soggetti affetti da questa sindrome hanno difficoltà nell'ultimare alcuni compiti complessi e questo problema può richiedere tempi più lunghi, meno

efficienza e più errori nelle attività rispetto al passato. I principali sintomi che si riscontrano riguardano difficoltà di memoria, problemi a concentrarsi o pianificare attività e problemi di linguaggio. A causa di quest'ampia varietà di sintomi, la patologia può essere classificata in base alla funzione cognitiva colpita in:

- *MCI amnesico*, quando nella persona è compromessa la sola memoria;
- *MCI non amnesico*, quando la persona presenta il deficit in un dominio cognitivo diverso dalla memoria, come ad esempio l'attenzione, il linguaggio o le funzioni esecutive;
- *MCI multidominio*, quando la persona presenta deficit in più funzioni cognitive.

Per aiutare i medici a individuare e valutare l'invecchiamento e la demenza, furono pubblicati, nel 1982, due metodi di valutazione, ancora oggi molto utilizzati. Questi metodi sono:

- il *punteggio clinico di demenza* o CDR, che permette di distinguere le persone sane da quelle in uno stadio di lieve demenza;
- la *scala globale di deterioramento per l'invecchiamento e la demenza* o GDS, che prevede sette stadi clinici, di cui quattro vanno dalla normalità alla lieve demenza.

Questi metodi permettono di valutare la gravità della patologia, ma non sono degli strumenti diagnostici.

Inizialmente, la diagnostica del MCI si basava principalmente sulla valutazione dei deficit nella memoria. Oggi, invece, si vanno a valutare anche un'ampia gamma di deficit cognitivi e non, tenendo conto anche dei fattori che possono influire sulle prestazioni cognitive, come l'educazione, i fattori di rischio vascolari, lo stato psichiatrico, il background genetico e la componente comportamentale e psicologica (come la depressione). Altri strumenti che nel futuro potrebbero aiutare l'identificazione della sindrome sono l'imaging e lo studio dei biomarcatori.

Studi hanno dimostrato che, su una popolazione di età superiore ai 65 anni, la percentuale di soggetti affetti da MCI è tra il 3% e il 19%. Circa il 44% di questi soggetti sono stabili e tornano alla normalità entro un anno, mentre tra l'11% e il 33% rischiano di sviluppare una qualche forma di demenza entro 2 anni.

Per prevenire e curare questa patologia, sono stati utilizzati, inizialmente, dei trattamenti farmacologici, in cui la durata del trattamento andava da

6 mesi a 3 anni. Un'analisi sul miglioramento cognitivo nei soggetti che assumevano questi farmaci ha dimostrato la loro infruttuosità. Infatti, la loro introduzione non ha portato a nessun miglioramento significativo sull'arresto della progressione verso la demenza.

Alcuni studi hanno rivelato, invece, che l'uso di un allenamento cognitivo su soggetti affetti da MCI, permetteva di ottenere dei risultati incoraggianti. Il maggior effetto è stato riscontrato su anziani sani, ma ci sono stati visibili miglioramenti anche nei pazienti con il morbo di Alzheimer.

Per la medicina è, perciò, molto importante identificare e trattare questi individui che soffrono di decadimento cognitivo lieve, in modo da prevenire o ritardare la progressione verso la demenza.

1.2 Malattie neurodegenerative

Le malattie neurodegenerative [7] sono un insieme di patologie caratterizzate da un'irreversibile e progressiva perdita di cellule neuronali in specifiche aree del cervello.

Il problema di base è dovuto al fatto che i neuroni non sono cellule che si riproducono, perciò quando subiscono dei danni o muoiono, questi non possono essere sostituiti dall'organismo. La morte delle cellule nervose può provocare disturbi motori o del funzionamento mentale.

Le cause di insorgenza di queste malattie sono ancora poco chiare, ma si ritiene che siano dovute a più fattori tra cui l'ereditarietà e fattori di tipo genetico e ambientale. Tendenzialmente, l'inizio della malattia è asintomatico e quando si manifestano i sintomi, il danno è già esteso. La progressione della malattia è irreversibile, dato che non esistono cure in grado di arrestare il degrado cognitivo, ma lo si può rallentare con un allenamento mirato.

Le malattie neurodegenerative principalmente conosciute sono: morbo di Alzheimer, morbo di Parkinson, sclerosi laterale amiotrofica, corea di Huntington e le demenze.

Morbo di Alzheimer È una delle patologie neurodegenerative più note. La perdita dei neuroni, dovuto a un accumulo anomalo di proteine (come la β-amiloide e la proteina Tau), causa delle gravi alterazioni alle funzionalità cognitive, incidendo principalmente sulla memoria a breve termine, sulle capacità di apprendimento e sulla sfera affettiva del paziente. La condizione del cervello in un paziente affetto da questo morbo è possibile vederlo in Figura 1.1a.

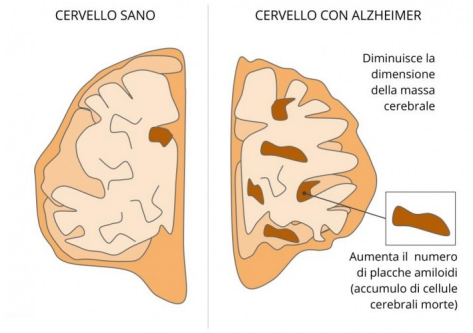
Morbo di Parkinson Caratterizzato dalla degenerazione dei neuroni della substantia nigra del cervello (area del cervello che controlla svariate funzioni motorie), che provocano la compromissione delle capacità di movimento del soggetto, come mostrato in Figura 1.1b. I principali sintomi riscontrabili sono: tremore a riposo, rallentamento nei movimenti (bradicinesia), rigidità muscolare e alterazione del bilanciamento posturale.

Sclerosi laterale amiotrofica (SLA) Patologia che riguarda i motoneuroni del sistema nervoso centrale, cioè i neuroni che controllano i muscoli (vedi Figura 1.1c). Il danneggiamento di tali cellule compromette tutti i movimenti (inclusa la respirazione) e provoca difficoltà a deglutire e deambulare, difficoltà nel linguaggio, fiato corto o alterato (dispnea) e insufficienza respiratoria.

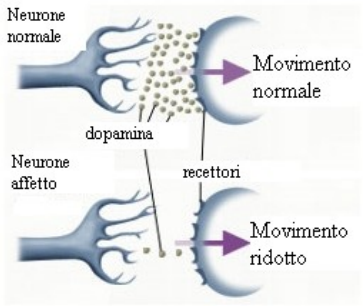
Corea di Huntington È una malattia ereditaria, che compromette le capacità cognitive e motorie. I sintomi più evidenti possono essere il cambiamento d'umore, la perdita di memoria, la difficoltà nel deambulare, parlare e deglutire e la depressione. La causa scatenante di questa patologia è la presenza, nel DNA del soggetto affetto, del gene della proteina huntingtina mutato (sono presenti più ripetizioni di triplette "CAG" del normale, come mostrato in Figura 1.1d).

Demenze Sono un gruppo di malattie tipiche dell'età avanzata, in cui il danno neuronale provoca un declino nelle funzioni cognitive. La sintomatologia varia in base all'area del cervello che è stata colpita. Le forme di demenza maggiormente diffuse, escludendo il morbo di Alzheimer già presentato, sono:

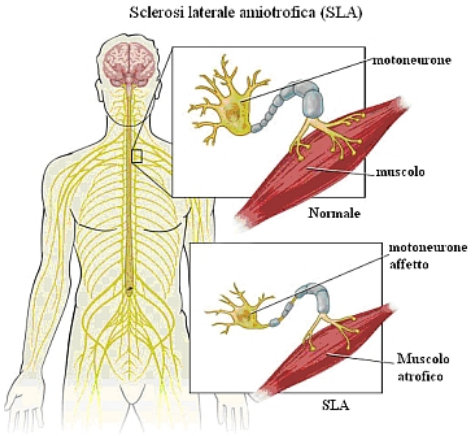
- la *demenza frontotemporale* [6], che colpisce i neuroni dei lobi frontali e temporali del cervello. I principali sintomi riscontrabili sono difficoltà nell'articolare un discorso, cambiamenti improvvisi nella personalità, disturbi nell'equilibrio e amnesie.
- la *demenza a corpi di Lewy* [5] (vedi Figura 1.1e), che consiste nella presenza all'interno del neurone di un agglomerato proteico insolubile, che blocca il rilascio di dopamina e acetilcolina, provocando il morbo di Parkinson e alcune forme di morbo di Alzheimer.
- la *demenza vascolare* [13] (vedi Figura 1.1f), che è causata da un'alterazione della circolazione sanguigna nel cervello, che priva alcune aree del cervello di sangue, provocando la morte dei neuroni. I sintomi di questa patologia dipendono dalla zona del cervello colpita dalla malattia.



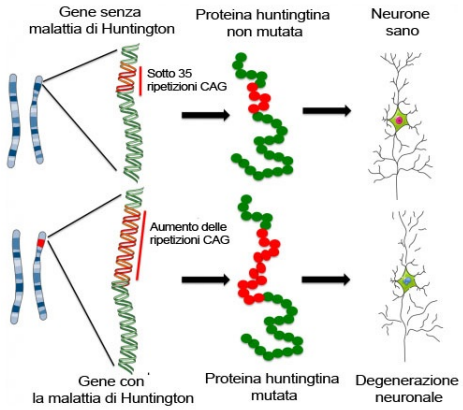
(a) Morbo di Alzheimer [3]



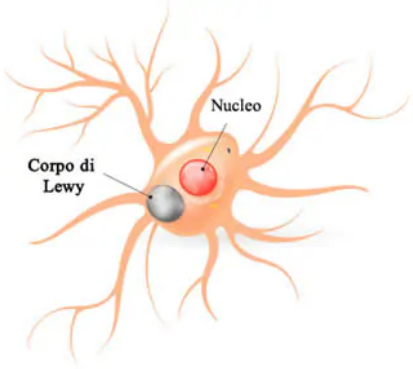
(b) Morbo di Parkinson [21]



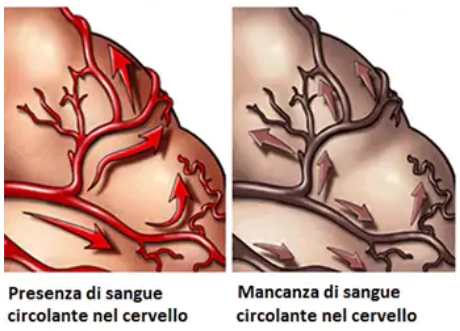
(c) SLA [22]



(d) Corea di Huntington [17]



(e) Demenza a corpi di Lewy



(f) Demenza vascolare

Figura 1.1: Rappresentazioni grafiche di come le malattie neurodegenerative più conosciute intaccano il corpo umano.

L'unico modo conosciuto per cercare di contrastare le malattie neurodegenerative è l'uso di farmaci specifici e l'impiego di strumenti che permettano di svolgere esercizi di memoria.

1.3 Training cognitivo computerizzato

Rispetto agli approcci tradizionali che prevedono l'uso di penna e matita, il training cognitivo computerizzato permette un allenamento multidominio, portando a un significativo miglioramento delle principali funzioni cognitive. Inoltre, prevedendo un intervento completamente personalizzato, aumenta la difficoltà del compito fruito al paziente in base alla sue capacità. Utilizzare il computer come strumento clinico prima dell'avvento della demenza, può portare a molti più benefici di un tradizionale trattamento farmacologico.

Il training cognitivo computerizzato prevede delle esercitazioni guidate riguardanti attività standardizzate e progettate per specifici processi cognitivi. Il compito da svolgere può appartenere a un singolo dominio cognitivo o a multipli domini.

Il training cognitivo computerizzato può essere catalogato in tre tipologie di allenamento informatizzato: training cognitivo classico, programmi software neuropsicologici e videogiochi. Per ogni tipologia sono stati effettuati degli studi [16] che hanno permesso di compararne i benefici ottenuti su pazienti adulti anziani, cioè che fanno parte di quella categoria di persone con un più alto rischio di declino cognitivo causato dall'invecchiamento, con capacità cognitive normali. Per poter comparare le tre tipologie di allenamento, sono stati presi in considerazione i seguenti aspetti:

- i *tempi di reazione*, cioè la quantità di tempo necessario per elaborare e rispondere a uno stimolo, fondamentale per il trattamento delle informazioni;
- la *velocità di elaborazione*, che è l'abilità di elaborare rapidamente le informazioni;
- la *memoria*, quindi la capacità di conservare, immagazzinare e richiamare informazioni;
- la *funzione esecutiva*, che comprende un'ampia gamma di abilità, inclusa la pianificazione, la flessibilità e la capacità di pensiero astratto;
- l'*attenzione selettiva*, cioè il processo svolto da un individuo quando si concentra su uno specifico stimolo nell'ambiente.

Training cognitivo classico Tipologia di allenamento dove il paziente allena specifiche abilità, come ad esempio la memoria o la velocità di elaborazione, utilizzando compiti standardizzati. Secondo alcuni studi effettuati, il tempo di reazione del paziente può, in alcuni casi, migliorare dopo l'allenamento. I benefici riportati da questo allenamento possono essere molteplici. Tra questi troviamo un miglioramento della velocità, delle abilità spaziali visive, del tempo di reazione, dell'attenzione, della memoria di lavoro, della funzione esecutiva e del ragionamento logico.

Programmi software neuropsicologici Tipo di training che permette di migliorare molteplici domini cognitivi utilizzando una grande varietà di compiti. Il partecipante può progredire in base al suo ritmo di apprendimento e ricevere dei feedback istantanei. L'utilizzo di questi programmi permette di migliorare la memoria episodica, l'apprendimento verbale, la velocità di elaborazione, la memoria uditiva, la percezione, il ragionamento e può portare benefici ai compiti percettivi su cui è stato o meno fatto l'allenamento. È meno efficace sull'attenzione e sulla funzione esecutiva.

Videogiochi Software dove il paziente deve manipolare le immagini per raggiungere un obiettivo. A differenza dei programmi software neuropsicologici, i videogiochi non sono stati progettati per migliorare le abilità cognitive, perciò non si focalizzano ad allenare uno specifico dominio cognitivo. I benefici, riscontrati dopo questo tipo di training, possono essere un miglioramento nei tempi di reazione, nelle abilità visive, nella funzione esecutiva, nella velocità di elaborazione, nella memoria di lavoro, nella memoria visiva a breve termine, nella capacità di ragionamento e nella velocità psicomotoria.

A seguito di questi studi, si può affermare che gli strumenti informatici permettono di ottenere miglioramenti nelle abilità cognitive. I risultati ottenuti sono paragonabili o anche migliori rispetto all'allenamento non computerizzato, perciò la loro adozione è giustificata.

L'anziano non avrà bisogno di essere tecnologicamente esperto per poter beneficiare dell'allenamento. Infatti, ricerche hanno dimostrato che avere già competenze informatiche non provocava un miglioramento significativo nell'apprendimento. Nonostante per molti anziani l'uso di tecnologie sconosciute provochi ansia, l'utilizzo di queste tipologie di training cognitivo porta ad alti livelli di soddisfazione nel soggetto.

Il training cognitivo computerizzato può essere applicato anche su anziani affetti da MCI o demenze.

Per constatare l'efficacia di questa tipologia di trattamenti anche su soggetti che soffrono di decadimento cognitivo lieve, sono stati svolti studi rag-

gruppati in meta-analisi [15]. Da queste ricerche è sorto che, su un campione di persone con età compresa tra i 60 e gli 80 anni, l'allenamento computerizzato è una modalità di somministrazione che migliora la cognizione delle persone affette da MCI e porta a risultati migliori rispetto a quelli ottenuti su anziani sani o affetti dal morbo di Parkinson. I principali miglioramenti che si notano riguardano la cognizione globale, la memoria, la memoria di lavoro e l'attenzione. L'uso di questi strumenti porta benefici anche all'umore dei pazienti, perciò si ha un miglioramento della funzione psicosociale e dei sintomi depressivi. In caso di demenza, invece, non si riscontrano dei miglioramenti significativi, ma può essere utile adottare tecnologie di realtà aumentata, che permettono un miglioramento nelle abilità visuo-spaziali.

Si può concludere che il training cognitivo ha effetti terapeutici per la categoria di anziani affetti da MCI, ma la sua efficacia risulta essere più limitata in presenza di demenza. I benefici riportati a seguito di un allenamento di questo tipo sono visibili fino a 3 mesi dopo la fruizione [14]. Ciò porta a pensare che è bene svolgere un allenamento duraturo nel tempo per poter migliorare le abilità cognitive, mantenere i benefici acquisiti e resistere al declino cognitivo. Se non si prevedere un allenamento continuo, le prestazioni della memoria diventano sempre più vulnerabili e l'allenamento sempre meno efficace.

1.4 Il progetto "Train your mind"

Il progetto "Train your mind" nasce dalla necessità del Rotary Club di Cesena di mettere a disposizione delle strutture mediche, nel territorio comunale, di un software per il training cognitivo.

L'obiettivo è quello di ottenere un software open source che permetta ai medici e neuropsicologi di svolgere un allenamento cognitivo su pazienti anziani sani o con condizioni patologiche più o meno gravi, per ritardare il loro declino cognitivo.

1.4.1 Rotary International

Il Rotary International [19] è un'associazione mondiale di imprenditori e professionisti, che prestano servizio umanitario, che incoraggiano il rispetto di elevati principi etici nell'esercizio di ogni professione e che si impegnano a costruire un mondo di amicizia e di pace.

Il primo Rotary Club fu fondato a Chicago nel 1905 dall'avvocato Paul P. Harris, il quale si incontrò con tre amici, di nazionalità e religioni diverse, per dar vita a un club di persone di differenti professioni, con lo scopo di

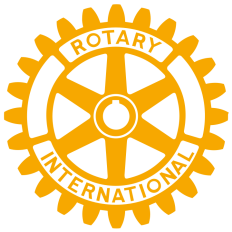


Figura 1.2: Emblema del Rotary International.

organizzare incontri regolari all'insegna dell'amicizia. A quel primo incontro presero parte Sylvester Schiele, un commerciante di carbone di origini tedesche, Gustav E. Loehr, un ingegnere minerario di origini svedesi, e Hiram E. Shorey, un sarto di origini irlandesi. Alla fine del 1905, il numero dei soci del club era salito a trenta componenti.

Il nome, Rotary Club, prende spunto dalle riunioni svolte dal club, le quali si tenevano a rotazione negli uffici dei vari soci. Il logo (mostrato in Figura 1.2) raffigura una ruota, simbolo dell'attività professionale e del tavolo conviviale a 8 posti (il numero perfetto della convivialità), con 24 denti (numero delle riunioni semestrali consigliate ai soci).

Il primo obiettivo del club fu aiutare le persone meno fortunate, perciò la prima attività benefica realizzata fu la costruzione di un servizio igienico pubblico nella piazza del municipio di Chicago.

Le cause umanitarie, a cui i progetti del Rotary si dedicano maggiormente, riguardano sei aree di intervento:

- *Promuovere la pace*: comprende progetti ideati per combattere problemi come la povertà, la disuguaglianza, le tensioni etniche, il bullismo e le violenze domestiche.

- *Combattere le malattie*: include progetti concepiti per fornire i mezzi per combattere malattie mortali e per prevenirle nelle comunità. Principalmente combattono contro l'HIV/AIDS, l'Alzheimer, la sclerosi multipla, il diabete e la Polio.

- *Forniture di acqua e strutture igenico-sanitarie*: comprende progetti volti a portare acqua pulita, servizi e igiene al maggior numero di persone, per ridurre anche la trasmissione delle malattie.

- *Proteggere madri e bambini*: annovera progetti pensati per migliorare le condizioni di vita di madri e bambini, fornendo un'assistenza sanitaria di maggior qualità.

- *Sostenere l'istruzione*: include progetti realizzati per sostenere l'alfabetizzazione e l'educazione di base, per ridurre le disparità di genere.

- *Sviluppare le economie locali*: annovera progetti creati per far crescere lo sviluppo economico e realizzare posti di lavoro ben retribuiti.

Dal 1979, una delle opere benefiche, a cui il Rotary ha maggiormente contribuito, è stata la lotta contro la Polio, con l'obiettivo di immunizzare 6 milioni di bambini nelle Filippine. Grazie all'intervento del Rotary, che ha contribuito a combattere questa malattia, oggi la Polio resta diffusa in 3 Paesi (Afghanistan, Pakistan e Nigeria), rispetto ai 125 nel 1988.

Oltre a svolgere un'azione internazionale nelle sei aree di intervento, ogni Rotary Club sviluppa anche progetti e iniziative per dare sostegno alle esigenze e necessità della comunità che abita nell'ambito del territorio del club. In questo modo, ogni rotariano può mettere in pratica la filosofia dell'associazione *"Servire al di sopra di ogni interesse personale"*.

Il Rotary Club di Cesena ha deciso di contribuire avviando un service riguardo alla salute e alla prevenzione delle malattie nel suo territorio. Per questo motivo, ha richiesto la realizzazione di questo progetto che andrà a combattere o ritardare l'invecchiamento cognitivo.

1.4.2 Soluzioni esistenti

Nell'ambito del training cognitivo computerizzato esistono già diverse soluzioni software fruibili in modo gratuito o a pagamento, tramite dispositivi mobili o desktop, in versione offline e online.

Alcuni esempi di software per il training cognitivo disponibili sono "Brainer"[1] (piattaforma web a pagamento pensata per i neuropsicologi), "Training cognitivo"[2] (sito web che mette a disposizione un'area di game center, dove si può scegliere tra giochi online e materiale da stampare) e "Lumosity"[3] (piattaforma web e app mobile con giochi per allenare la mente, in parte a pagamento).

I principali problemi riscontrati nei programmi già esistenti sono i seguenti:

- parte degli esercizi previsti sono molto generali, perciò non permettono di concentrare l'allenamento su uno specifico dominio cognitivo;

[1] http://www.brainer.it
[2] http://www.trainingcognitivo.it
[3] https://www.lumosity.com

- gli esercizi spesso non permettono al paziente di fare elaborazioni proprie, in quando vengono sempre presentate le opzioni tra cui scegliere la risposta. Ciò porta a non allenare bene le funzioni esecutive dei pazienti;
- i software, in alcuni casi, non permettono una buona user experience;
- in alcuni programmi mancano le spiegazioni degli esercizi o delle soluzioni errate.

Proprio per via dei problemi riscontrati nei software per l'allenamento cognitivo già esistenti, il committente ha preferito optare per una nuova soluzione, che possedesse come elemento di novità degli esercizi mirati su specifiche funzionalità cognitive e che permettesse un'estensione del sistema nel futuro.

Capitolo 2

Analisi dei requisiti

Per progettare il sistema richiesto, in modo che soddisfi i bisogni del committente, è necessario svolgere una fase preliminare per definire i requisiti di sistema. I requisiti indicano cosa il cliente richiede ad un sistema, cioè lo scopo del sistema software, senza però definire come sarà costruito.

La raccolta dei requisiti è stata eseguita con lo svolgimento di una serie di interviste. Le interviste sono state realizzate direttamente sugli utenti finali, cioè ai dottori del Centro Studi e Ricerche in Neuroscienze Cognitive di Cesena.

L'obiettivo finale di "Train your mind" è quello di ottenere un sistema, in cui medici e pazienti possano registrare, monitorare e svolgere esercizi per il training cognitivo. Attraverso questo strumento, deve essere possibile gestire e personalizzare le sessioni di allenamento e tenere sotto controllo i risultati ottenuti dai singoli pazienti.

2.1 Requisiti funzionali

I requisiti funzionali sono quei requisiti che specificano le interazioni tra il sistema e l'ambiente. Vanno perciò a definire le funzioni che il sistema possiederà.

Per prima cosa sono stati definiti gli attori, cioè gli utenti esterni che il sistema deve supportare. Gli attori individuati sono i seguenti:

- il *medico*, cioè un qualsiasi operatore coinvolto nella riabilitazione neuropsicologica del paziente (il neurologo o psicologo che raccomanda l'uso del training, altre figure sanitarie supervisionate da questi oppure anche i caregivers che aiutano il paziente nell'uso del software a casa), che registra i pazienti, monitora le loro attività e personalizza il loro allenamento;

- il *paziente*, cioè il soggetto con un declino cognitivo, che svolge la sessione di training e deve portare a termine i compiti affidati;
- lo *sviluppatore*, cioè colui che sviluppa nuovi compiti per l'allenamento cognitivo.

In seguito, sono stati identificati tutti i casi d'uso, mostrati in Figura 2.1, i quali hanno permesso di individuare i requisiti funzionali del sistema. Tali requisiti sono, qui di seguito, descritti.

Gestione del profilo paziente Il sistema deve permettere all'utente di creare e gestire la scheda clinica del paziente. Per ogni paziente, si devono poter inserire tutte le informazioni sulla sua condizione fisica e psichica, per far avere al medico il suo quadro clinico completo. Tra le informazioni del paziente devono comparire i dati anagrafici, l'educazione, i medicinali assunti, le patologie che lo affliggono, il consumo di alcolici, il tipo di manualità e se è un fumatore.

Gestione del profilo di esercitazione Il sistema deve permettere all'utente di creare e gestire la scheda di un profilo di esercitazione. Ogni profilo di esercitazione dovrà includere una lista di esercizi, che il paziente dovrà svolgere durante le sessioni di allenamento, se il profilo gli verrà assegnato.

Assegnazione di esercizi al profilo L'utente deve poter assegnare una lista di esercizi a un profilo di esercitazione. Gli esercizi assegnati devono poter comparire anche più volte all'interno della stessa esercitazione e devono poter essere svolti con un ordine predefinito, quando richiesto.

Assegnazione del profilo di esercitazione al paziente Una volta registrati il profilo e il paziente, si deve poter assegnare il profilo di esercitazione creato al paziente, in modo che quest'ultimo possa poi svolgere delle esercitazioni con gli esercizi assegnati a quel profilo.

Oltre all'assegnazione, deve poter essere possibile rimuovere il profilo assegnato al paziente, quando non gli si vuole più far svolgere quel tipo di esercitazione.

Selezione dei livelli di difficoltà degli esercizi di un profilo Dopo aver assegnato il profilo di esercitazione a un paziente, il medico deve poter selezionare i livelli di difficoltà di default per ogni esercizio presente nel profilo

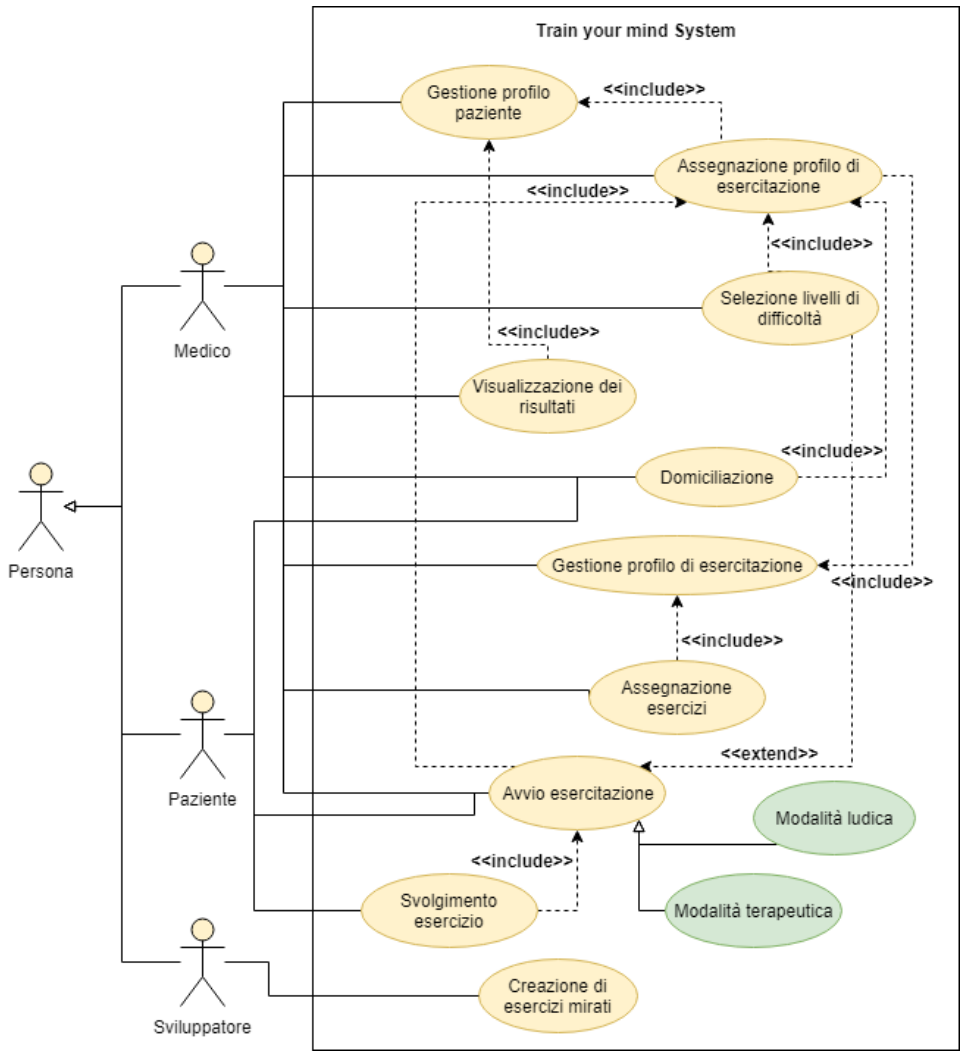


Figura 2.1: Diagramma dei casi d'uso del sistema "Train your mind".

assegnato. Ciò significa che, all'avvio dell'esercitazione, ogni esercizio sarà eseguito con il livello selezionato in precedenza.

Inoltre, il sistema deve dare la possibilità al medico di modificare tali livelli di difficoltà anche prima dell'avvio dell'allenamento.

Avvio dell'esercitazione Una volta assegnato il profilo di esercitazione a un paziente, si deve poter avviare e svolgere l'esercitazione. L'esercitazione dovrà essere composta dagli esercizi assegnati al profilo, che dovranno essere svolti con un ordine ben preciso, se era previsto l'ordine obbligatorio.

Le modalità di svolgimento dell'esercitazione devono essere di due tipologie:

- *Modalità terapeutica*: tipologia di allenamento dove, al termine dell'esercitazione, devono essere salvati i risultati ottenuti;
- *Modalità ludica*: categoria di training, durante il quale, il paziente può testare gli esercizi, ma i risultati ottenuti non saranno memorizzati e non andranno ad incidere sul suo andamento globale.

Svolgimento di un esercizio Il paziente deve poter avviare in autonomia l'esercizio da svolgere. Prima dell'avvio dell'esercizio, devono essere disponibili delle istruzioni che ne spiegano il funzionamento.

Una volta avviato l'esercizio, l'utente, oltre a svolgerlo come richiesto, deve poter tornare alle istruzioni in caso di necessità e deve poter tenere sotto controllo il tempo rimanente.

Inoltre, si deve dare anche la possibilità di uscire dall'esercizio per poterlo ricominciare. Questa modalità può essere utile nei casi in cui il medico si accorge che la consegna non è stata compresa a pieno o che il paziente non era concentrato.

Al termine dell'esercizio, il software deve dare un feedback all'utente, indicando se la risposta data è corretta o errata. Nel caso di risposta errata, si deve mostrare la soluzione corretta. Infine, dovranno essere segnalati i risultati ottenuti in termini di tempo impiegato e di risposte corrette o errate.

Visualizzazione dei risultati Il medico deve avere la possibilità di analizzare i risultati ottenuti da ogni singolo paziente nei singoli esercizi. I risultati devono essere fruibili in una qualche modalità, che permetta all'utente di analizzarli in modo semplice, veloce, ma anche approfondito. Sarà quindi utile analizzare i dati in base al tempo impiegato dal paziente per svolgere l'esercizio, il numero di errori commessi, il livello di difficoltà assegnato all'esercizio svolto e la categoria di appartenenza dell'esercizio.

Domiciliazione Il sistema deve permettere al medico di domiciliare gli esercizi al paziente, in modo che quest'ultimo li possa svolgere anche dalla propria abitazione, senza doversi recare alla struttura.

Creazione di esercizi mirati per specifiche funzionalità cognitive Il sistema deve permettere allo sviluppatore di creare esercizi mirati per l'allenamento di specifiche funzionalità cognitive. Gli esercizi creati dovranno essere aggiunti al framework degli esercizi in modo semplice e standardizzato. Gli esercizi che dovranno essere inseriti nel sistema sono i seguenti:

- matrici attentive;
- SART;
- attenzione alternata;
- N-back;
- position N-back;
- Stop signal;
- ordinamento crescente di numeri interi;
- combinare la figura al dettaglio.

2.2 Requisiti non funzionali

I requisiti non funzionali sono quelle proprietà misurabili dall'utente e non correlate al comportamento funzionale del sistema.

Per questo progetto, i requisiti non funzionali individuati sono tre e sono esposti qui di seguito.

Utilizzo di tecnologie note e standard Il sistema che si vuole ottenere in output deve essere open source, in modo che possa essere esteso da chiunque voglia implementare nuove funzionalità o esercizi. È stato quindi richiesto di utilizzare tecnologie standard e note alla maggior parte degli sviluppatori, in modo che si possa facilitare l'estensibilità del progetto.

Interfaccia user-friendly L'interfaccia del software da realizzare deve essere facile da usare, permettendo all'utente di avvalersi del software in autonomia, senza bisogno di una specifica formazione. L'utente dovrà anche essere in grado di ricavare e interpretare i risultati ottenuti dai singoli pazienti.

Realizzazione di un sistema facilmente estensibile e modificabile Il sistema che si vuole realizzare deve essere progettato in modo che possa essere facilmente estensibile e modificabile da chiunque ne abbia la necessità, senza provocare delle modifiche a catena nel codice. In questo modo, nel futuro, sarà possibile aggiungere dei nuovi esercizi al framework oppure prevedere un cambiamento personalizzato, in base al paziente, della skin del programma.

Capitolo 3

Architettura e progettazione

Definire l'architettura del software prima di implementare il codice è una buona pratica per costruire un modello che permetta di mostrare come il sistema dovrà essere strutturato e i componenti che ne andranno a far parte. Le decisioni prese in questa fase del progetto hanno avuto un impatto decisivo sullo sviluppo del lavoro.

L'architettura è l'organizzazione del sistema, in termini di componenti e relazioni tra i componenti e l'ambiente. Attraverso l'architettura si definiscono anche i principi che guidano la progettazione e l'evoluzione del software, perché le scelte fatte sugli aspetti non visibili all'utente finale vanno a vincolare i singoli elementi del progetto. L'obiettivo è quindi quello di velocizzare la realizzazione delle singole funzionalità del sistema, cercando di soddisfare tutti i requisiti.

3.1 Struttura architetturale

L'architettura è stata definita nella prima fase di progettazione, con lo scopo di scomporre il sistema in sottosistemi.

L'obiettivo di questa fase è stato quello di predisporre il sistema in modo che la sua futura conversione da sistema stand-alone a sistema distribuito fosse semplificata. Per rispettare questo obiettivo è stata scelta come tipologia architetturale quella *client-server*, perché è stata ritenuta la più idonea alle esigenze del sistema "Train your mind".

La progettazione del sistema centralizzato ha portato ad ottenere un'architettura composta da tre componenti, mostrati in Figura 3.1, che comunicano tra loro per scambiarsi i dati. Questi componenti sono:

- un *client*, che è il componente che accede ai servizi offerti dal server e interagisce direttamente con l'utente finale;

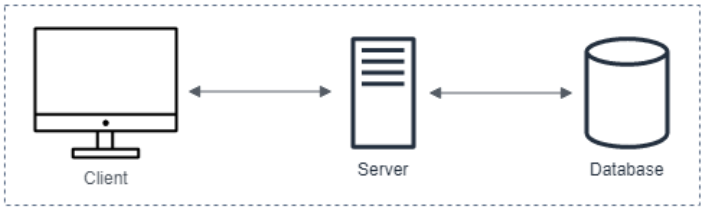


Figura 3.1: Visione grafica complessiva dell'architettura del sistema.

- un *server*, che è il componente che si occupa della gestione logica del sistema, fornendo servizi al client e svolgendo operazioni sul database;
- un *database*, che è il componente preposto alla memorizzazione dei dati. Vi può accedere solamente il server, al quale comunica le informazioni in base alle richieste ricevute.

La gestione della concorrenza tra i tre componenti e la progettazione del database saranno affrontati nel seguito della trattazione.

3.1.1 Organizzazione in moduli

Un modulo è un componente del sistema software che è legato ad altri moduli attraverso delle relazioni. La scomposizione in moduli ha l'obiettivo di ridurre la complessità di un progetto, di supportare la modificabilità, l'estensibilità e il riutilizzo.

Grazie all'organizzazione in moduli del progetto è stato possibile mantenere separati i singoli componenti del sistema e gestire al meglio l'interazione tra di essi. I moduli definiti, però non corrispondono esattamente ai tre componenti individuati durante la progettazione della struttura architetturale. I moduli che compongono il progetto sono i seguenti:

- modulo `client`: modulo che contiene l'applicazione del client. In esso sono presenti tutti i layout delle view e i meccanismi per far funzionare l'applicazione;
- modulo `server`: modulo contenente i servizi offerti dal server e il database;
- modulo `utils`: modulo che comprende gli oggetti utilizzati all'interno del progetto, i messaggi per le comunicazioni tra i componenti e il framework degli esercizi.

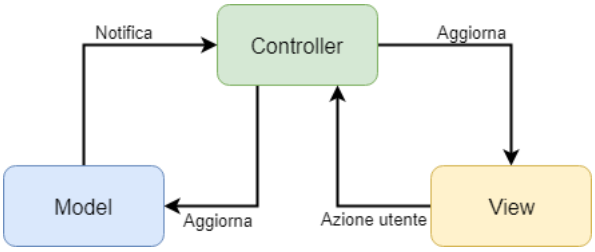


Figura 3.2: Struttura del pattern MVC con la tipica interazione tra i componenti.

3.1.2 Scelta dei pattern architetturali

L'ultima operazione svolta per la progettazione della struttura architettonica è stata quella di individuare i pattern architetturali da applicare al progetto.

Un pattern architetturale descrive il modello organizzativo strutturale di un sistema software in termini di sottosistemi e di relazioni tra essi. Offre, quindi, una soluzione generale a un problema che si verifica comunemente all'interno di un'architettura. La selezione di pattern architetturali per il progetto è una decisione fondamentale nella progettazione di un sistema software. Rispetto ai design pattern, il pattern architetturale opera a un livello più alto ed esprime schemi che permettono di impostare l'organizzazione strutturale di un sistema.

I pattern architetturali scelti per il progetto sono: Model-View-Controller e Data Access Object.

Model-View-Controller

Il pattern Model-View-Controller (o MVC) [10] è un pattern che permette di separare la logica di presentazione dei dati dalla logica di business.

Il MVC, la cui struttura è mostrata in Figura 3.2, permette di dividere l'applicazione in tre componenti:

- il *model*, in cui sono contenuti e gestiti i dati e che descrive la logica del programma;
- la *view*, con cui si mostrano le informazioni all'utente attraverso delle interfacce grafiche;
- il *controller*, che è situato tra il model e la view, perciò gestisce gli input dell'utente e fornisce le informazioni necessarie al model.

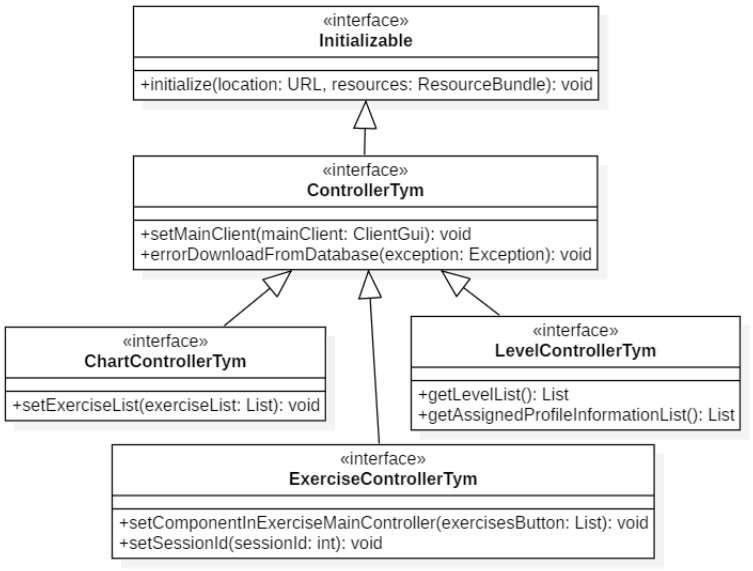


Figura 3.3: Diagramma delle classi con la gerarchia delle interfacce dei controller del progetto.

Il vantaggio che porta la divisione in componenti dell'applicazione è che ogni singolo componente può essere riutilizzato, modificato e sostituito, senza che queste operazioni vadano ad incidere sugli altri.

Il pattern MVC è, quindi, stato applicato al progetto corrente, per poterne trarre tutti i benefici. Esso si comporrà di:

- model formati dagli oggetti che implementano l'interfaccia `TymModel`;
- view realizzate in formato FXML, per via dell'adozione di JavaFX;
- controller, uno per ciascuna view, che implementano una delle interfacce della gerarchia mostrata in Figura 3.3.

Data Access Object

Il pattern Data Access Object (o DAO) [4] è un pattern utilizzato per separare i servizi dell'application layer dalle operazioni di accesso ai dati. Questa separazione permette di nascondere completamente i dettagli di implementazione della sorgente dati e di incapsulare l'accesso ai dati in un livello separato.

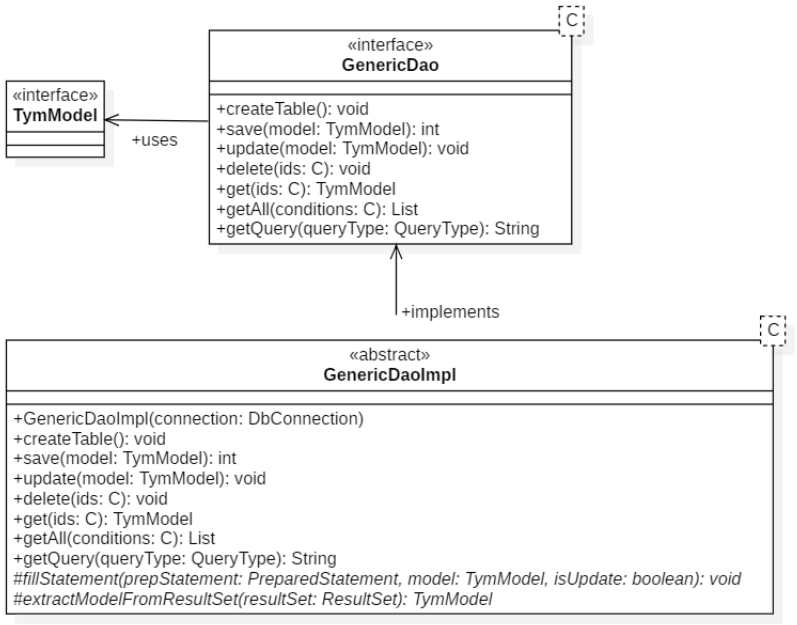


Figura 3.4: Struttura del pattern DAO del progetto "Train your mind".

L'interfaccia esposta dal DAO non cambia in base all'origine dei dati sottostante, in modo da adattarsi a diversi schemi di memorizzazione senza dover modificare nulla.

Il compito del DAO è gestire la connessione con l'origine dati per permettere la memorizzazione o il trasferimento dei dati. L'origine dati può essere di una qualsiasi tipologia, come ad esempio RDBMS o XML.

La struttura del pattern DAO, che verrà implementato all'interno del progetto, è mostrata in Figura 3.4. Ci sarà un'interfaccia `TymModel`, che modella un qualsiasi oggetto del sistema. Questa interfaccia sarà utilizzata da un'interfaccia `GenericDao`, che definisce tutte le operazioni standard da eseguire sugli oggetti nell'origine dati. L'implementazione dell'interfaccia `GenericDao` avverrà attraverso una classe astratta `GenericDaoImpl`, la quale implementa le operazioni comuni a tutte le tabelle nella sorgente dati, mentre lascia ai singoli DAO delle tabelle l'implementazione di altri metodi, che hanno un funzionamento più specifico.

3.2 Progettazione del database

I dati raccolti durante l'utilizzo del software, devono rimanere persistenti per poter essere utilizzati o analizzati anche in momenti futuri. È necessario creare una base dati in cui memorizzare le informazioni in modo strutturato.

Un database è un archivio di dati organizzati secondo un criterio logico. Conoscere la struttura dei dati permette di inserirli e modificarli in modo rapido.

La tipologia di database scelta per il progetto è quella relazionale, perché attualmente è la più popolare e conosciuta. La decisione è stata presa sempre nel rispetto del requisito non funzionale di utilizzare tecnologie note e standard. Il modello relazionale si basa sull'uso di tabelle, dove vengono posizionate le informazioni, correlate tra loro tramite l'algebra relazionale.

Il primo passo per la realizzazione del database è stato scegliere quale sistema di gestione di database relazionali (RDBMS) utilizzare. La prima scelta è ricaduta su MySQL [18], il RDBMS open source più utilizzato, perché è molto veloce, affidabile, scalabile e facile da usare; inoltre, è in grado di gestire database di grandi dimensioni. Il problema principale è che per poter poi eseguire l'applicazione, si sarebbe dovuto installare un gestore del database su ogni elaboratore utilizzato. Questa modalità è molto scomoda per un progetto che attualmente può essere eseguito solo in locale, ma diventerà una buona soluzione quando sarà trasformato in un progetto distribuito.

L'idea è, quindi, quella di utilizzare un database embedded, cioè una tecnologia in cui la gestione del database è incorporata all'interno dell'applicazione, anziché essere un database stand-alone. Il codice sarà però strutturato in modo che un eventuale modifica alla tipologia di database adottata, non provochi modifiche a catena sulla struttura delle tabelle nel database.

L'RDBMS utilizzato è Apache Derby [9], un database relazionale open source scritto in Java, che può essere incorporato direttamente nei programmi scritti in quel linguaggio. Il suo problema principale, è che in confronto a MySQL è più lento. Questa pecca può essere accettata, in quanto non va ad intaccare la fluidità del software. Si prevede in sviluppi futuri di sostituirlo quando si migrerà alla versione distribuita.

A seguito della scelta del RDBMS, è stata definita la struttura dei dati e le tabelle necessarie per rappresentare tutte le informazioni. La struttura del database è visibile in Figura 3.5. Le tabelle definite sono:

- `Doctor`: tabella che racchiude tutte le informazioni che riguardano un medico. Nella versione attuale del database non sarà utilizzata, in quanto si prevede che su una singola applicazione ci lavori un solo medico.

- `Patient`: tabella che gestisce i dati dei pazienti.
- `Drug taken`: tabella contenente le informazioni riguardanti i medicinali assunti dai pazienti. Ogni medicinale inserito appartiene a uno specifico paziente.
- `Illness`: tabella che comprende i dati delle patologie dei pazienti. Come per i medicinali, anche ogni patologia inserita è associata a uno specifico paziente.
- `Exercise`: tabella che contiene tutti i dati di un esercizio. Per avere uno storico degli esercizi inseriti negli allenamenti, questi non verranno rimossi, ma si indicherà se l'esercizio non è più attivo.
- `Exercise profile`: tabella che gestisce le informazioni riguardanti i profili di esercitazione.
- `Exercise in profile`: tabella necessaria per indicare gli esercizi inseriti all'interno dei profili di esercitazione.
- `Assigned exercise profile`: tabella che racchiude i dati dell'assegnamento di un profilo di esercitazione a un paziente.
- `Exercise in assigned profile`: tabella in cui sono contenuti gli esercizi del profilo di esercitazione al momento dell'assegnamento, compreso il livello di default che il medico assegna a ogni esercizio per uno specifico paziente.
- `Exercise session`: tabella contenente le informazioni di ogni sessione di esercitazione svolta per uno specifico profilo assegnato.
- `Exercise performed`: tabella in cui sono inseriti gli esercizi svolti nelle sessioni di esercitazione, con i relativi risultati ottenuti.

3.3 Gestione della concorrenza

Una volta scelta la tipologia di architettura e progettato il database, si è passati a progettare la gestione della concorrenza e la comunicazione tra il client e il server.

Il paradigma ad oggetti è un paradigma di programmazione che è stato rivoluzionario fin dalla nascita, ma ha sempre avuto un problema per quanto riguarda la concorrenza. Per ovviare a questa lacuna, è stata introdotta la

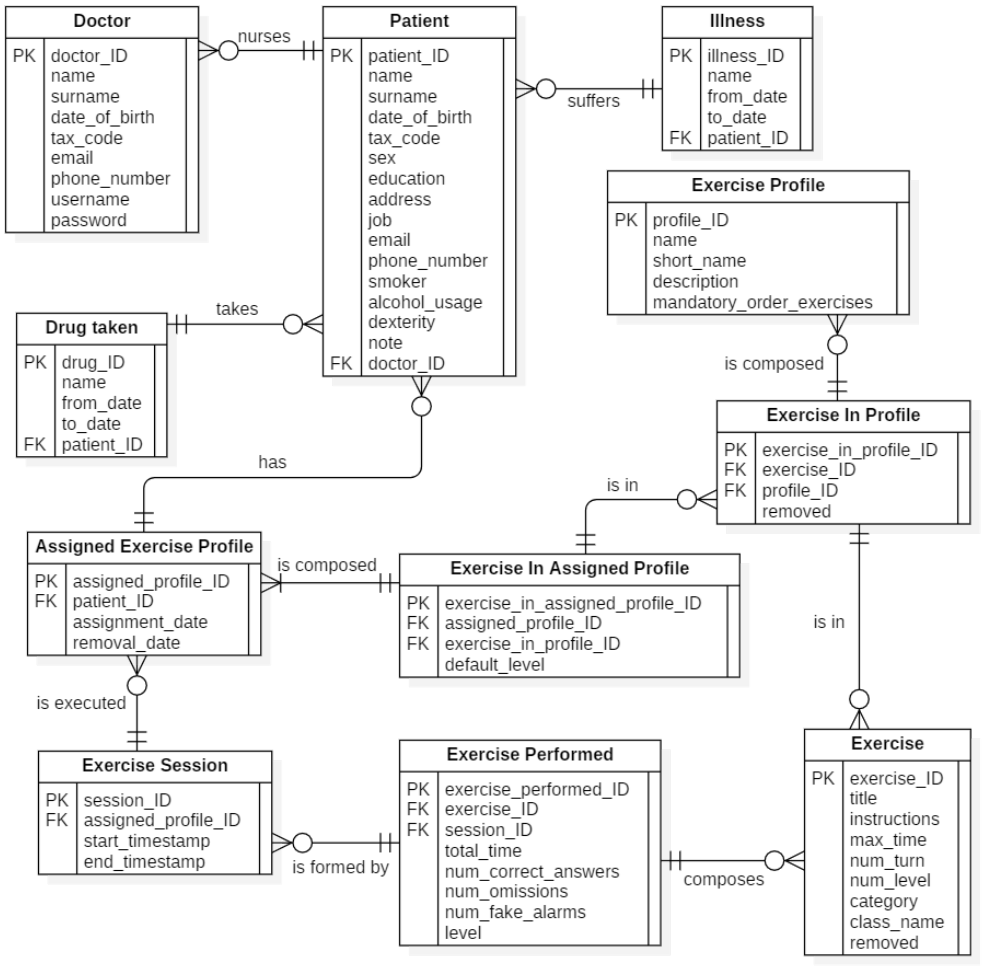


Figura 3.5: Struttura dei dati nel database del progetto “Train your mind”.

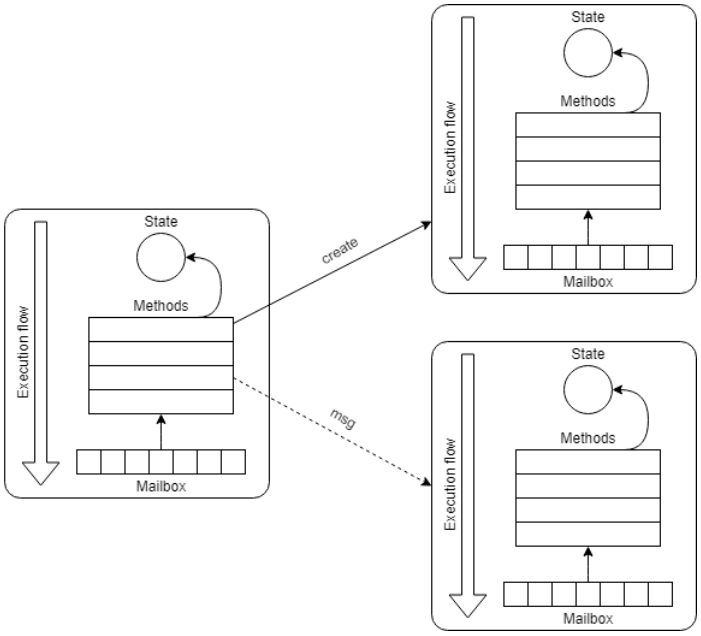


Figura 3.6: Rappresentazione semplificata del concetto di attore.

programmazione multithread, che ha permesso il parallelismo tra i processi all'interno delle applicazioni. Questo tipo di programmazione ha, però, alcuni problemi nella comunicazione, che portano a limitarne le performance. Questo succede perché, avendo ogni thread un flusso di esecuzione separato, devono essere svolte attività di sincronizzazione e coordinazione per poter far cooperare tra loro i singoli thread.

Per agevolare la programmazione concorrente è stato introdotto il *modello ad attori* [1], in cui la comunicazione avviene attraverso lo scambio di messaggi in modo asincrono tra entità chiamate attori. Nel modello ad attori, l'idea di base è che *"everything is an actor"*, dove un attore è un'entità autonoma che opera in modo concorrente rispetto agli altri attori, in un sistema del tutto asincrono. Un attore, la cui struttura è mostrata in Figura 3.6, ha al suo interno un proprio flusso di esecuzione (indipendente dagli altri), una mailbox dove raccoglie i messaggi in arrivo (l'arrivo è indeterminato per via dell'asincronismo) e il proprio stato interno. Lo stato interno non è condiviso con gli altri attori, perciò per venirne a conoscenza si dovrà inviare un messaggio all'attore interessato.

I principali vantaggi dell'adozione di questo modello sono:

- la separazione del controllo (il "dove" e il "come") dalla logica della computazione, permettendo la decomposizione del programma in

componenti autonomi (molto vantaggioso per i sistemi distribuiti);

- la semplificazione della gestione della coordinazione tra le applicazioni.

Tenendo sempre ben presente di voler progettare un sistema che permetta una pratica e veloce conversione a sistema distribuito, è stato deciso di gestire la concorrenza attraverso il modello ad attori. Tra tutte le varie implementazioni del modello esistenti, è stato scelto di utilizzare Akka.

Il *framework Akka* [2] è una libreria open source per la progettazione di sistemi scalabili e fault-tolerant basata sul modello ad attori. È scritto in Scala, ma integrabile anche in Java. Per via della sua capacità di gestire i fallimenti, di usare costrutti di concorrenza non di basso livello, di effettuare una comunicazione trasparente tra i componenti, di avere elasticità e scalabilità su richiesta, tale implementazione è diventata una tra le più popolari per il modello ad attori.

All'interno del progetto, grazie all'introduzione di Akka, si prevede l'utilizzo di tre oggetti:

- `Launcher`: oggetto che si occupa di creare il sistema degli attori e gli attori stessi.
- `ClientActor`: l'attore del client, il quale si occupa di avviare l'applicazione, di inviare le richieste al server tramite messaggi appositi e di gestire le risposte ricevute dal server.
- `ServerActor`: l'attore del server, il quale deve gestire le tabelle sul database, gestire le richieste del client, svolgere le operazioni sul database e inviare le risposte al client, in base al risultato ottenuto.

La tipologia di comunicazione adottata è di tipo RPC-like, in cui il mittente del messaggio (nel nostro caso è sempre il client) deve attendere la ricezione della risposta da parte del destinatario (il server) per poter procedere con altri eventuali messaggi. La Figura 3.7 schematizza il tipico funzionamento che deve possedere il sistema, mostrando la creazione del sistema di attori e un esempio di comunicazione tra l'attore del client e l'attore del server, comprendendo anche l'interazione del server con il database.

3.4 Progettazione dell'interfaccia grafica

La progettazione dell'interfaccia grafica nel progetto "Train your mind" è stata uno dei punti centrali del progetto, in quanto è ciò con cui l'utente finale si rapporta per poter interagire con il programma.

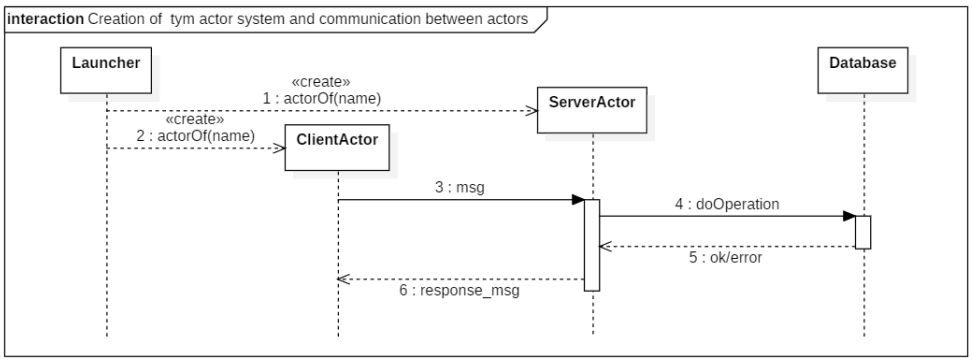


Figura 3.7: Schematizzazione del funzionamento degli attori del progetto "Train your mind".

Lo scopo della progettazione è stato quello di ottenere come risultato un'interfaccia che fosse facile da usare, semplice e intuitiva, ma che permettesse di svolgere anche le operazioni più avanzate di utilità medica. È stato perciò fondamentale focalizzarsi sulla user experience; infatti, senza una buona esperienza per l'utente, ci può essere un alto rischio di abbandono dell'uso del software e un ritorno ai metodi tradizionali per il training cognitivo.

La progettazione si è composta di due fasi: per prima cosa sono stati realizzati i wireframe e poi i mockup.

Il wireframe è una bozza del lavoro che dovrà essere svolto, dove viene mostrato lo scheletro dell'interfaccia. Tale bozza è stata realizzata su carta ed è diventata la base per realizzare i mockup.

Il mockup è una rappresentazione statica del prodotto finale, che contiene un alto livello di dettaglio e fedeltà. Serve per mostrare all'utente cosa si prevede di ottenere come risultato finale, senza però avere l'interattività. I mockup realizzati sono poi stati mostrati al committente per avere la sua approvazione o feedback su eventuali modifiche e per procedere con la fase di implementazione dell'interfaccia.

La struttura della navigazione della GUI è stata organizzata per tab, per permettere all'utente di muoversi da una sezione a un'altra del software in modo semplice e veloce. Le tab previste sono tre: la sezione dei pazienti, la sezione dei profili di esercitazione e la sezione delle esercitazioni. Ognuna di queste sezioni verrà analizzata qui di seguito.

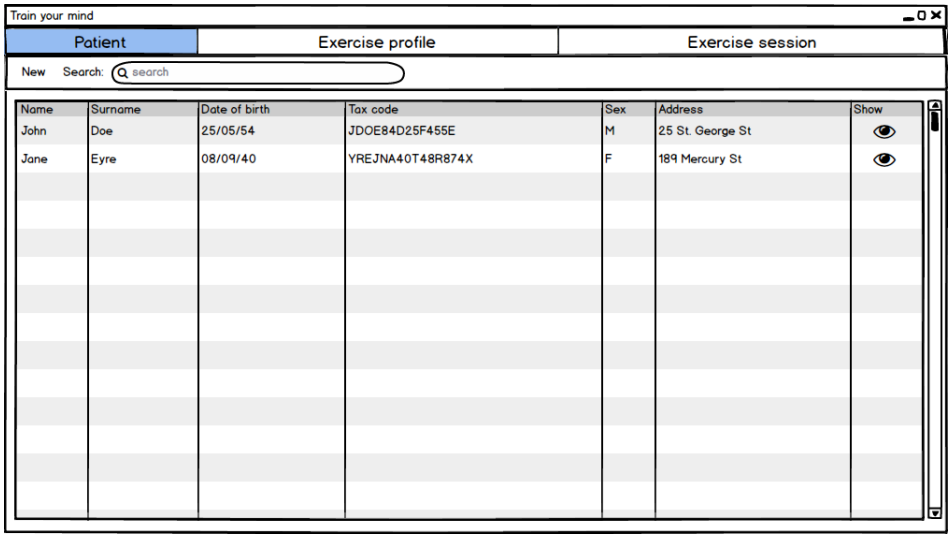


Figura 3.8: Mockup della home della sezione dei pazienti.

3.4.1 Sezione dei pazienti

Nella sezione dei pazienti si prevede di gestire i singoli pazienti, attraverso le loro schede cliniche, i profili di esercitazione assegnati e prendendo visione dei risultati ottenuti nelle singole sessioni di esercitazione.

Nella pagina principale di questa sezione, visibile in Figura 3.8, saranno elencati tutti i pazienti che il medico ha inserito e che sta tenendo in cura. Da questa schermata sarà possibile svolgere due tipi di azioni: inserire la scheda clinica di un nuovo paziente e gestire i singoli pazienti.

La Figura 3.9 mostra le dialog che permetteranno al medico di inserire i dati per creare la scheda clinica del paziente. Si prevede, quindi, di inserire i dati anagrafici, i recapiti e tutte le informazioni cliniche che possono essere utili per analizzare il caso specifico di ciascun paziente, come per esempio il livello di istruzione, la manualità, il consumo alcolico, i medicinali assunti o le patologie di cui soffre. Nell'elenco dei medicinali assunti e delle patologie è possibile aggiungere un nuovo dato clinico oppure rimuoverne uno già esistente attraverso i controlli che saranno presenti al di sopra di ogni elenco. Al termine dell'inserimento di tutte le informazioni, l'utente potrà decidere se salvare i dati inseriti oppure se annullare l'operazione.

Per ogni paziente sarà, poi, possibile vedere la sua scheda clinica, gestirne i profili di esercitazione assegnatigli e monitorare i risultati ottenuti nelle singole esercitazioni o esercizi svolti. La scheda clinica si organizzerà in due sezioni: la sezione in alto dove sarà presente un header, in cui saranno

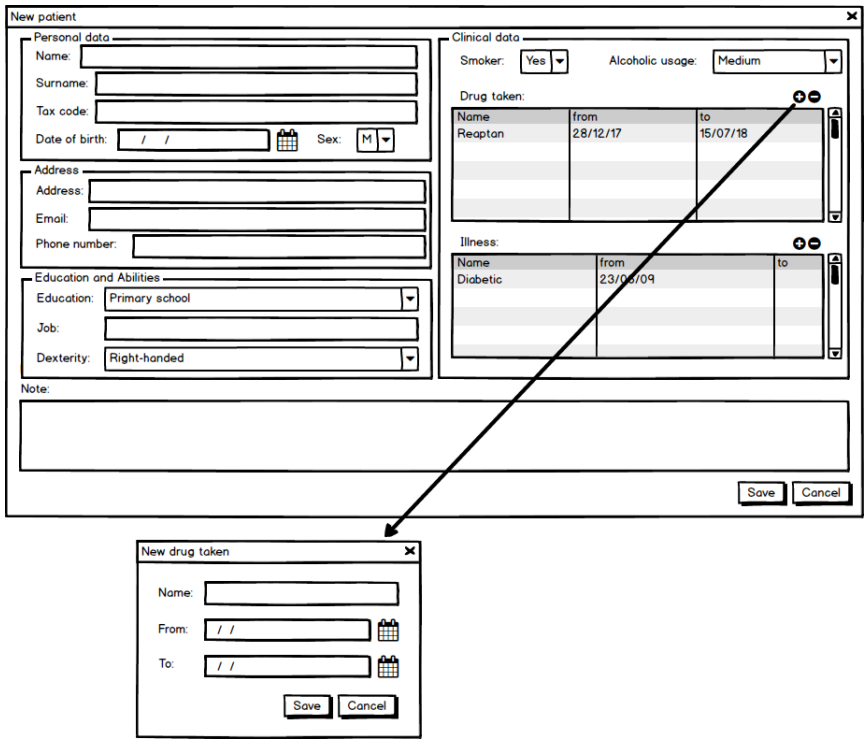


Figura 3.9: Mockup che mostra le dialog per l'inserimento della scheda clinica di un nuovo paziente.

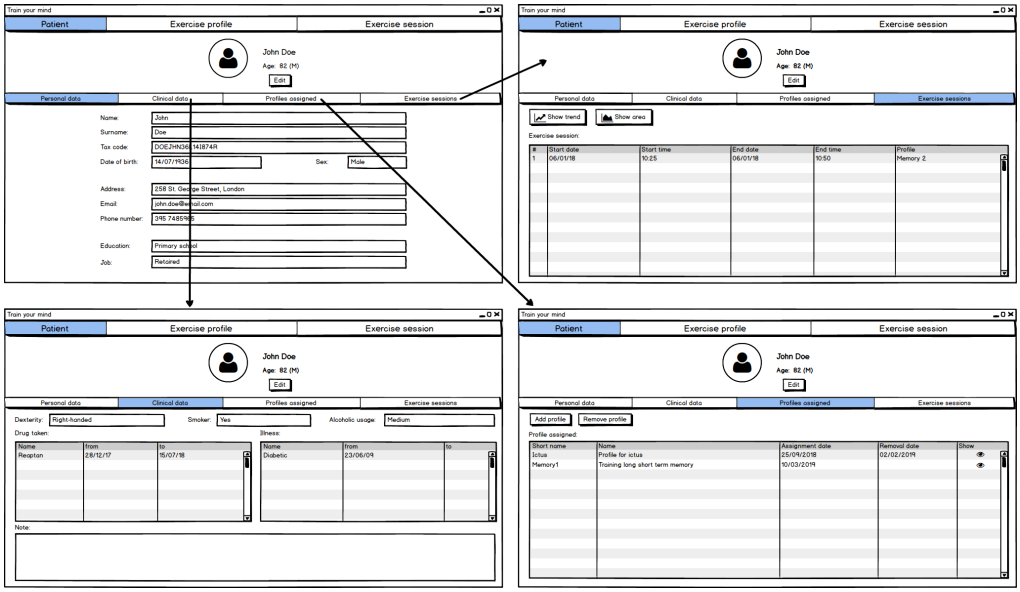


Figura 3.10: Mockup della scheda clinica di un paziente. Da qui si può accedere ai suoi dati personali, ai dati clinici, ai profili assegnati e alle informazioni sulle sedute di training.

mostrati i dati più rilevanti del paziente (come nome e cognome, età e sesso), e la sezione in basso dove sarà possibile navigare tutti i dati del paziente, tramite dei tab, come mostrato in Figura 3.10.

Per gestire i profili di esercitazione assegnati al paziente, è stata predisposta una sezione specifica nella scheda del paziente (Figura 3.11). Per ogni paziente, si potrà decidere di assegnargli un nuovo profilo oppure rimuovergli un profilo assegnato in precedenza. Quando si assegnerà al paziente un nuovo profilo, verrà mostrata una dialog in cui nell'elenco a sinistra saranno disponibili tutti i profili di esercitazione non ancora assegnatigli e non attivi, mentre nell'elenco a destra ci sono tutti i profili che gli si vorrà assegnare. In basso, invece, verranno mostrati i dati principali del profilo selezionato, per mostrare all'utente la descrizione e gli esercizi presenti al suo interno.

In ogni profilo di esercitazione assegnato, sarà possibile gestire i livelli di difficoltà degli esercizi presenti. Grazie a questa gestione, il medico sarà in grado di assegnare il giusto livello di difficoltà all'esercizio che il paziente dovrà svolgere. La gestione dei livelli, però, non potrà più essere eseguita, se il profilo assegnato è stato rimosso.

L'utente potrà monitorare i progressi del paziente nella sezione mostrata in Figura 3.12. Qui saranno elencate tutte le sessioni di esercitazione svolte,

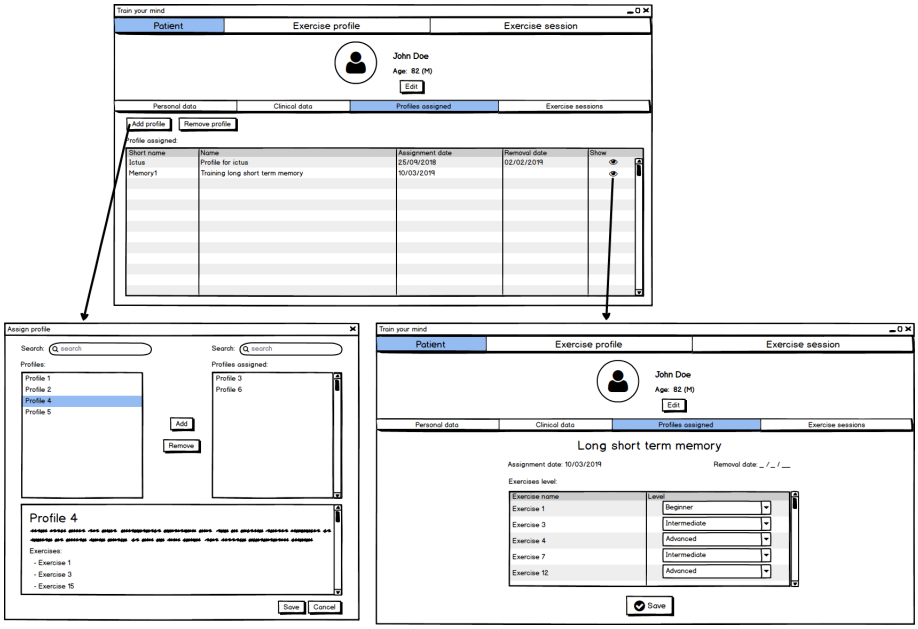


Figura 3.11: Mockup che mostra l'assegnazione di un profilo di esercitazione a un paziente e la gestione dei livelli di difficoltà degli esercizi.

dove per ognuna è possibile controllare gli esercizi svolti e per ogni esercizio il tempo impiegato, il livello di difficoltà e gli errori commessi.

Sarà possibile anche monitorare l'andamento del paziente nel tempo in base all'esercizio svolto. In questo caso, si potranno comparare i tempi impiegati per arrivare al termine dell'esercizio, gli errori commessi e i livelli di difficoltà assegnati all'esercizio. L'arco di tempo analizzato potrà essere filtrato in base al numero di mesi che si vogliono controllare oppure al numero delle ultime sessioni svolte.

Per monitorare l'andamento generale del paziente in base alla categoria dell'esercizio, cioè all'area cognitiva allenata, sarà messo a disposizione un radar chart.

3.4.2 Sezione dei profili di esercitazione

Nella sezione dei profili di esercitazione, mostrata in Figura 3.13, si vuole permettere all'utente di creare e gestire i profili di esercitazione, in modo del tutto personalizzato.

La pagina principale di questa sezione prevede un elenco con tutti i profili creati dall'utente. Ogni profilo avrà la propria scheda, nello stesso stile

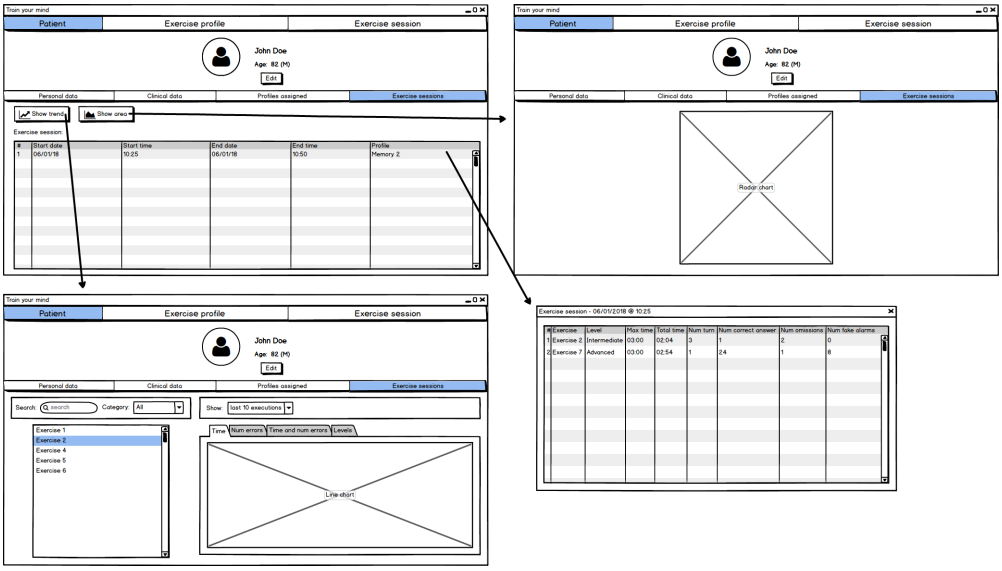


Figura 3.12: Mockup della scheda per il monitoraggio dei risultati ottenuti dal paziente. Da qui è possibile prendere visione degli esercizi svolti in ogni seduta e dei grafici di andamento nel tempo di ogni singolo esercizio svolto.

della scheda clinica del paziente, dove saranno elencate tutte le informazioni riguardanti il profilo e l'elenco degli esercizi assegnati.

La creazione di un nuovo profilo richiederà all'utente di inserire le informazioni di base del profilo, come ad esempio un titolo e una descrizione, e di assegnarci gli esercizi. Le caratteristiche per l'assegnazione degli esercizi sono le seguenti:

- ogni esercizio può essere assegnato più volte allo stesso profilo;
- l'utente può riordinare la lista degli esercizi assegnati in base alle necessità;
- l'utente può decidere di dare un ordinamento obbligatorio agli esercizi, perciò quando si avvierà l'allenamento con quel profilo, gli esercizi si potranno eseguire solo nell'ordine prestabilito.

3.4.3 Sezione delle sessioni di esercitazione

La sezione delle sessioni di esercitazione permette di avviare le sessioni di allenamento per i singoli pazienti su uno specifico profilo di esercitazione. Il suo funzionamento si divide in due operazioni: impostazione dell'esercitazione ed esecuzione dell'esercitazione precedentemente impostata.

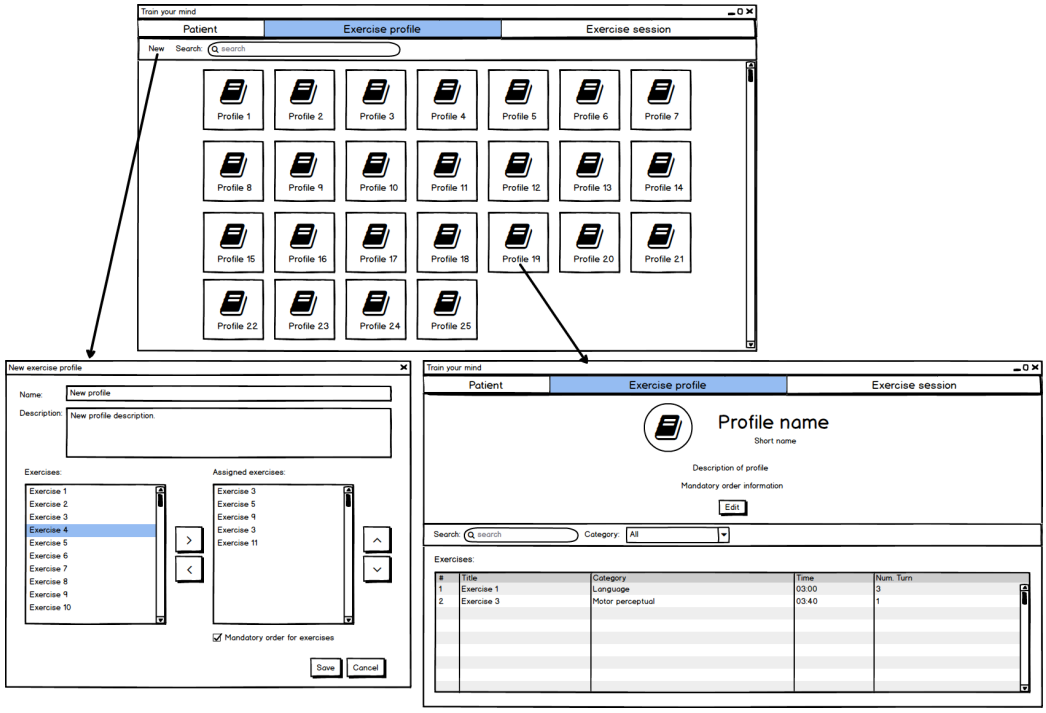


Figura 3.13: Mockup che mostra la gestione dei profili di esercitazione. In alto è visibile la home dei profili, in basso a sinistra la dialog per inserire un nuovo profilo e in basso a destra la scheda del singolo profilo.

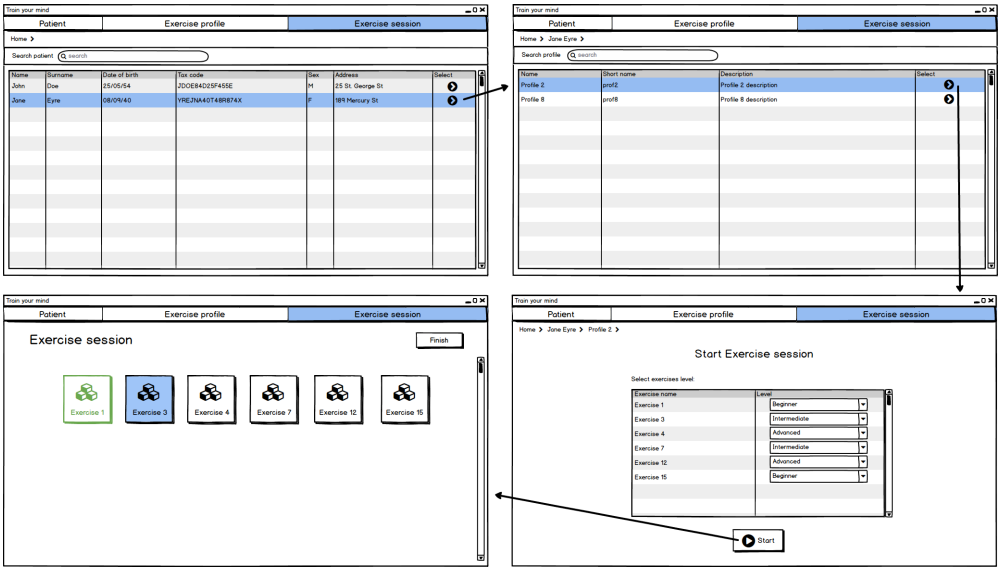


Figura 3.14: Mockup che mostra l'avvio di una sessione di esercitazione, in cui si seleziona il paziente, il profilo su cui fare l'allenamento e i livelli di difficoltà.

Impostare l'esercitazione significa che si dovrà permettere di selezionare il paziente a cui far eseguire l'allenamento, selezionare il profilo di esercitazione su cui si vuole svolgere l'allenamento (i profili mostrati saranno solo quelli assegnati al paziente e attivi), impostare i livelli di difficoltà dei singoli esercizi (cosa che si potrà fare anche nella scheda clinica del paziente) e poi avviare l'esercitazione. Le GUI per lo svolgimento di queste operazioni sono visibili in Figura 3.14.

All'avvio dell'esercitazione, l'utente potrà scegliere tra gli esercizi proposti quello da avviare (come mostrato in Figura 3.15), a meno che l'ordine non sia obbligatorio, in quel caso solo un esercizio sarà attivo per poter essere svolto. Una volta selezionato l'esercizio, verrà mostrata una dialog contenente le istruzioni dell'esercizio e il tempo massimo a disposizione per svolgerlo.

La schermata dell'esercizio sarà composta da due componenti: una barra degli strumenti e una sezione centrale dove verrà mostrato il compito. La barra sarà composta da:

- il nome dell'esercizio;
- un pulsante per accedere nuovamente alle istruzioni dell'esercizio;
- il tempo impiegato per svolgere l'esercizio;

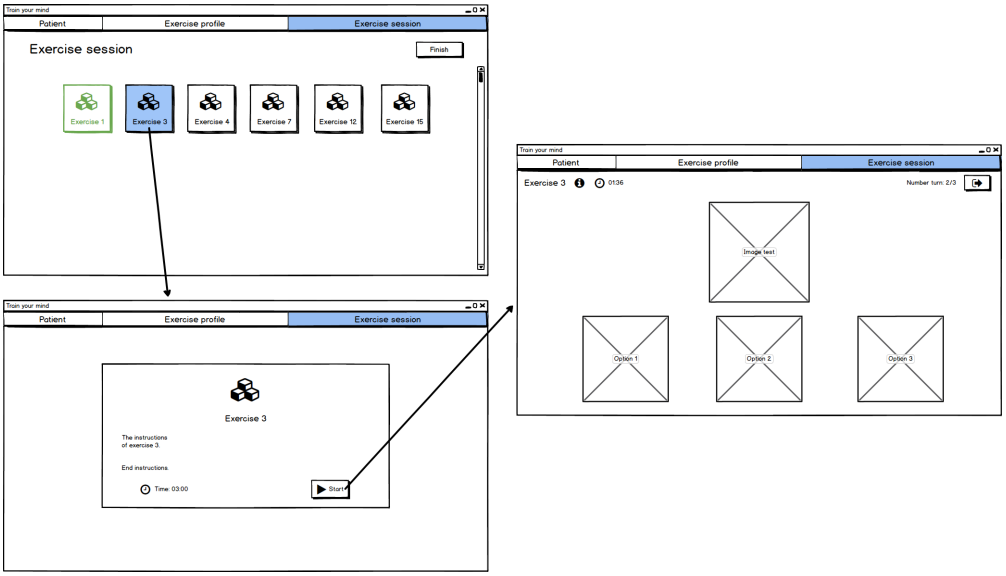


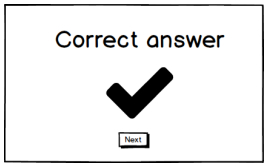
Figura 3.15: Mockup per mostrare l'avvio di un esercizio.

- il numero di turni che ha l'esercizio;
- un pulsante per uscire dall'esercizio e annullare tutti i progressi fatti. Utile per quando il medico ha la necessità di far ricominciare l'esercizio, soprattutto quando il paziente non ha prestato la giusta attenzione o si è compreso che il paziente non ha capito il funzionamento dell'esercizio.

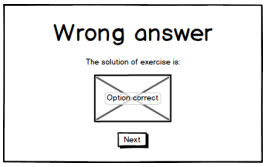
Al termine di ogni turno di un esercizio sarà rilasciato un feedback all'utente, dove si indicherà se la risposta data era corretta (Figura 3.16a). In caso di risposta errata, invece, dovrà essere mostrato, come da richiesta del committente, la soluzione corretta, in modo che l'utente possa capire gli errori commessi (Figura 3.16b). Invece, al termine dell'esercizio, si dovrà mostrare una dialog contenente i risultati ottenuti in termini di tempo impiegato, di numero di risposte corrette e numero di risposte errate (Figura 3.16c).

Il committente, per questa sezione, si è focalizzato su due richieste molto importanti:

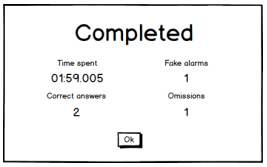
- durante l'esercitazione, il computer dovrà essere lasciato al paziente per poter svolgere l'esercizio, perciò questo soggetto non dovrà avere la possibilità di spostarsi su altre schede o modificare i dati di competenza del medico;
- si dovrà dare la possibilità al medico di scegliere se vuole salvare i dati dell'esercitazione appena svolta. In questo modo, si è in grado di



(a) Dialog per la risposta corretta.



(b) Dialog per la risposta errata.



(c) Dialog per i risultati finali dell'esercizio.

Figura 3.16: Insieme di mockup che mostrano le possibili dialog che compaiono durante lo svolgimento di un esercizio.

permettere lo svolgimento dell'esercitazione in modalità terapeutica e ludica.

Per soddisfare la prima richiesta, si dovrà disabilitare la navigazione a tab tra le tre sezioni principali. Così facendo, il paziente avrà la possibilità di operare solamente all'interno dell'ambiente che si crea all'avvio della sessione di training.

Per la seconda richiesta, si dovrà mettere a disposizione un pulsante che permetta di uscire dalla sessione di esercitazione. Durante l'uscita, verrà richiesto al medico se vuole salvare i dati raccolti. Se si salvano i dati, allora l'esercitazione è svolta in modalità terapeutica, altrimenti sarà svolta in modalità ludica. La modalità ludica serve al paziente per testare gli esercizi e capirne il funzionamento, oppure si verifica quando il medico ritiene che l'esercitazione appena svolta non sia stata conclusa con il giusto grado di attenzione.

3.4.4 Localizzazione

La localizzazione del software (detta anche *l10n*) è l'insieme di processi di adattamento e traduzione di un software alle esigenze linguistiche, culturali, tecniche e legali proprie di un certo ambiente. Grazie alla localizzazione si permette agli utenti finali di accedere al software nella loro lingua nativa, superando le barriere culturali e raggiungendo, così, un pubblico molto più ampio.

Per il progetto "Train your mind" si è deciso di progettare un'interfaccia utente localizzata, astraendo tutti gli elementi localizzabili (come ad esempio i testi visibili all'utente) e posizionandoli in file esterni di cui si potranno avere le diverse traduzioni.

Questa decisione è stata presa, perché il Rotary opera in tutto il mondo e in questo modo si permette l'utilizzo del software in una qualsiasi lingua, semplicemente avendo a disposizione la traduzione.

Attualmente la lingua disponibile per il progetto è l'inglese, ma si prevede di realizzare una traduzione italiana nel futuro, per permettere l'utilizzo del software all'interno di tutte le strutture mediche sul territorio locale.

Nel progetto sarà creato un file esterno in cui si associa ad ogni chiave una stringa di testo. Attraverso una classe statica, chiamata `BundleUtils`, si potrà ottenere il testo appropriato da applicare ad un certo componente grafico.

3.5 Design di dettaglio dell'applicazione

A seguito della progettazione dell'interfaccia grafica è stato possibile procedere con la progettazione dei componenti che devono comporre l'applicazione.

Il design di dettaglio ha richiesto due passi principali:

- la definizione della struttura del modello;
- la definizione delle viste e dei controller.

3.5.1 Modello

Il modello è stato progettato in modo che contenesse tutti i metodi di accesso agli oggetti del sistema. In totale si sono previste dodici classi, che modellano i singoli oggetti, e due generalizzazioni, le quali permettono di modellare la porzione di dati in comune a più classi. Entrando più nello specifico, possiamo raggruppare l'identificazione degli oggetti del modello in tre gruppi:

- per poter gestire i pazienti, come richiesto dal committente, si ha la necessità di avere un oggetto `Patient`, che permetta di modellare i dati di un paziente, e uno `Doctor`, il quale modella i dati di un medico curante. Entrambi questi oggetti sono persone, perciò avranno informazioni in comune che saranno modellate con un oggetto `Person`. Per ogni paziente, poi, si devono poter gestire i medicinali assunti e le patologie di cui soffre. Per rappresentare queste due informazioni si utilizzano gli oggetti `DrugTaken` e `Illness`, ma dato che entrambi sono dei dati clinici,

i dati in comune sono gestiti dall'oggetto `ClinicalData`. La struttura di questo gruppo di modello è visibile nel diagramma di Figura 3.17.

- per poter gestire i profili di esercitazione, invece, è necessario identificare un oggetto `ExerciseProfile`, il quale ne modella i dati. Dato che ad ogni profilo devono essere associati un insieme di esercizi, si deve definire anche l'oggetto `Exercise` e l'oggetto `ExerciseInProfile`, che permette di ricavare tutti gli esercizi assegnati ad un profilo. Una volta definito il profilo, dovrà essere data la possibilità di assegnare il profilo al paziente. Per modellare questo concetto saranno necessari due ulteriori oggetti: `AssignedExerciseProfile`, per gestire i dati dei profili assegnati ai pazienti, e `ExerciseInAssignedProfile`, per organizzare i dati degli esercizi nei profili assegnati. La rappresentazione grafica di queste relazioni sono visibili in Figura 3.18.

- per poter gestire le sessioni di allenamento saranno necessari due ulteriori oggetti (vedi Figura 3.19). Il primo è `ExerciseSession`, che permette di modellare i dati riguardanti la sessione di esercitazione, mentre il secondo è `ExercisePerformed`, il quale gestisce i dati di un esercizio svolto.

Per poter realizzare il modello in modo efficiente, è stato deciso di adottare il pattern Builder per la costruzione degli oggetti. L'utilizzo di questo pattern creazionale ha permesso di separare la costruzione di un oggetto complesso dalla sua rappresentazione. Un esempio del suo utilizzo è possibile vederlo nel diagramma di Figura 3.20.

In alcuni casi, dove si era in presenza di una gerarchia di oggetti, è stato necessario definire un builder astratto per la superclasse, il quale è esteso dai builder dei singoli oggetti. Questi builder implementano solamente i metodi astratti della superclasse, i quali permettono di ottenere l'interfaccia della classe e la classe del builder dell'oggetto. Un esempio di applicazione del pattern Builder in caso di gerarchia è visibile in Figura 3.21, dove si mostra l'applicazione del Builder per i dati clinici. Un'altra implementazione simile è stata fatta per le persone.

3.5.2 Viste e controller

A seguito della progettazione del modello, sono stati progettati le viste e i corrispondenti controller, i quali ne gestiscono il funzionamento. Nella pratica, i controller definiti andranno ad aggiornare le viste ad essi associate, in base alle interazioni dell'utente con la vista, e si serviranno dei modelli, precedentemente definiti, per svolgere le operazioni sui dati.

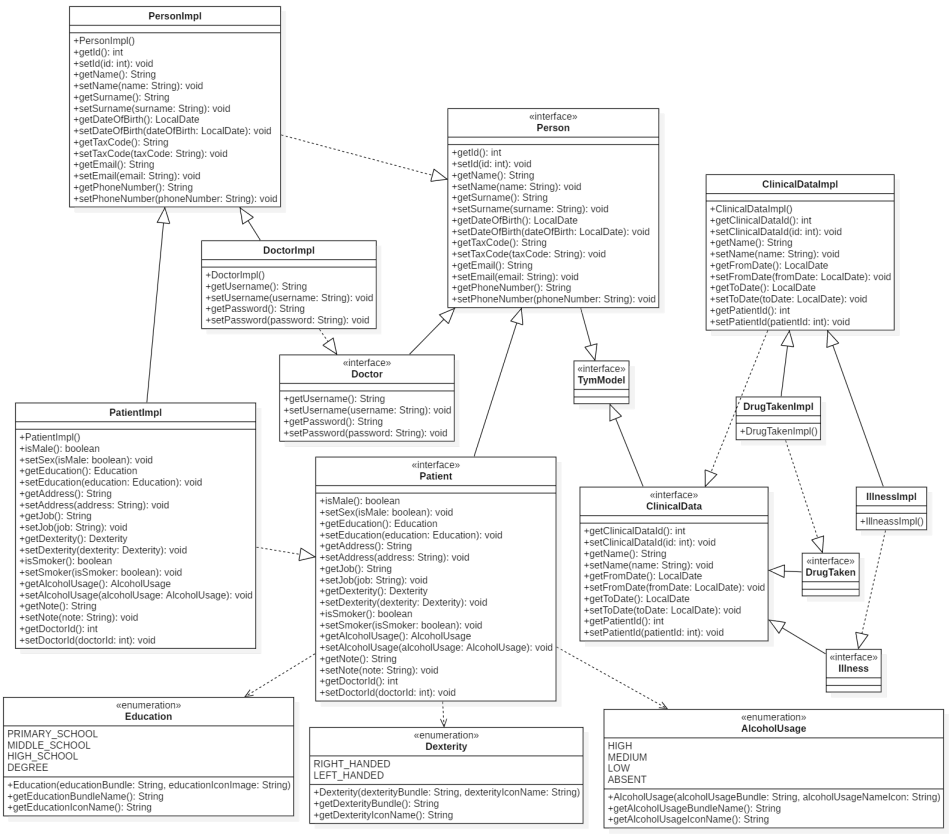


Figura 3.17: Diagramma delle classi con la struttura del modello per manipolare la gestione dei pazienti.

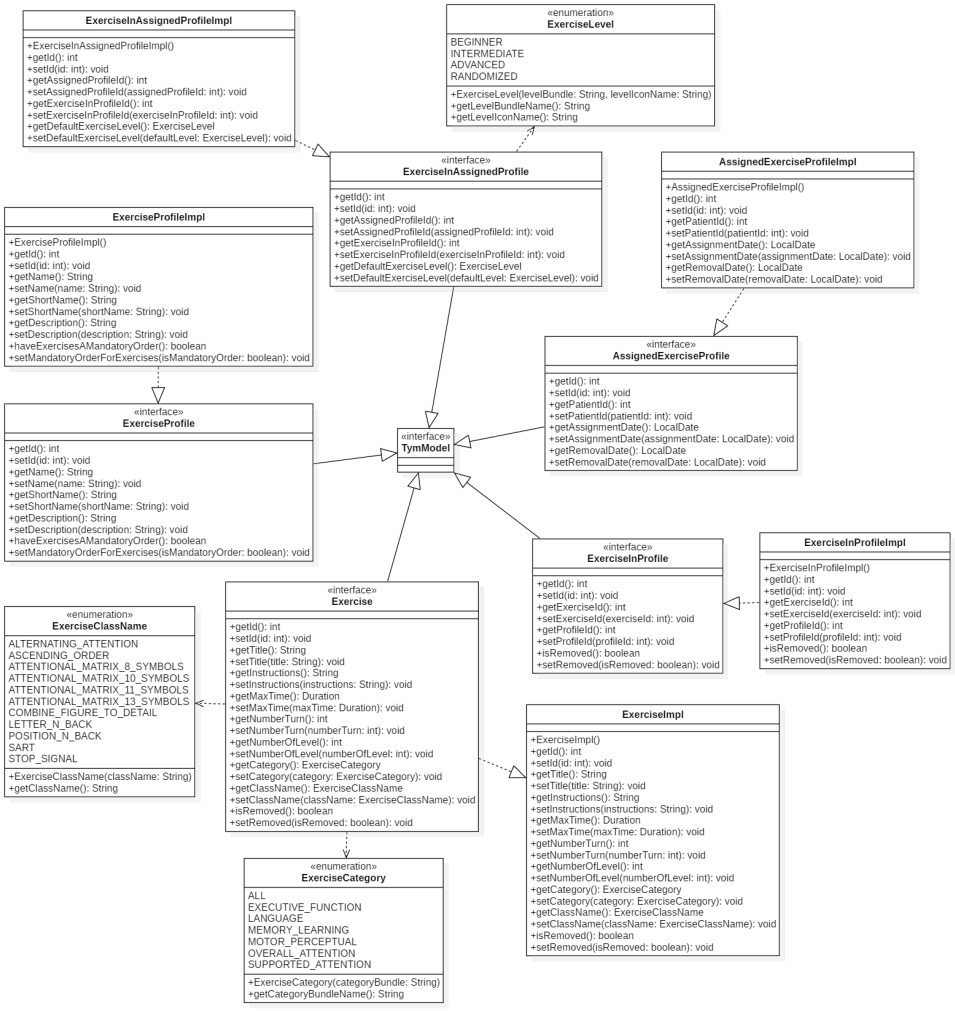


Figura 3.18: Diagramma delle classi con la struttura del modello per manipolare la gestione dei profili di esercitazione.

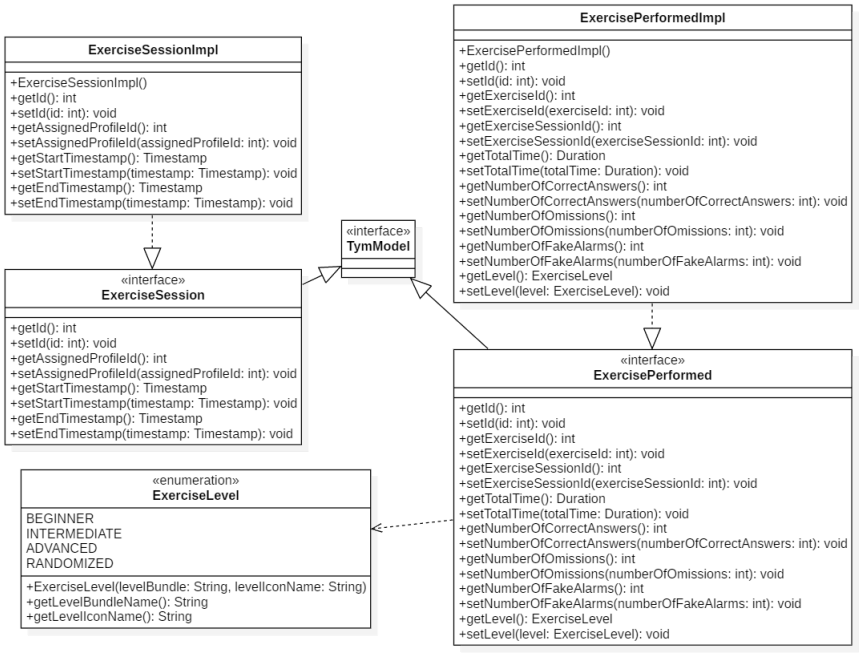


Figura 3.19: Diagramma delle classi con la struttura del modello per manipolare la gestione delle sessioni di esercitazione.

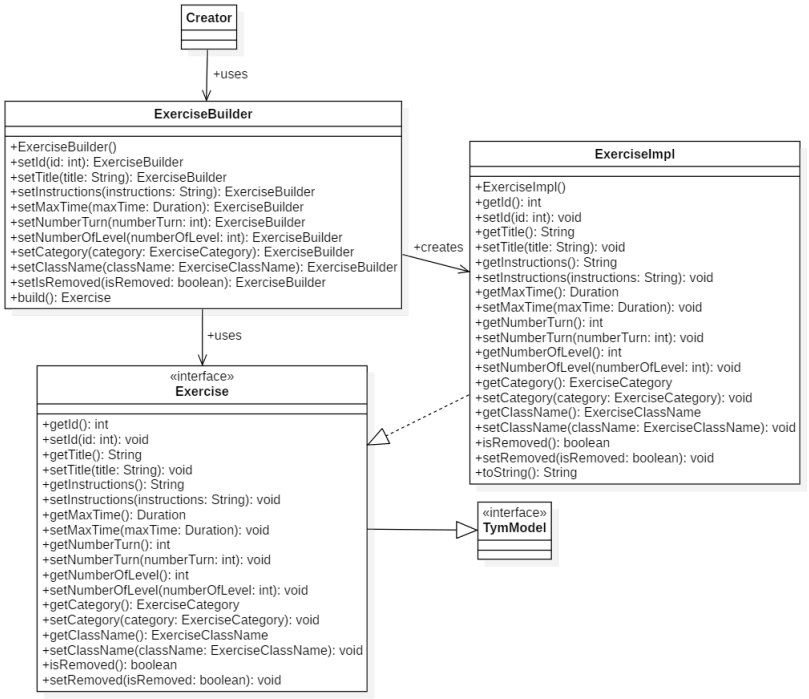


Figura 3.20: Diagramma delle classi del pattern Builder della classe `Exercise`.

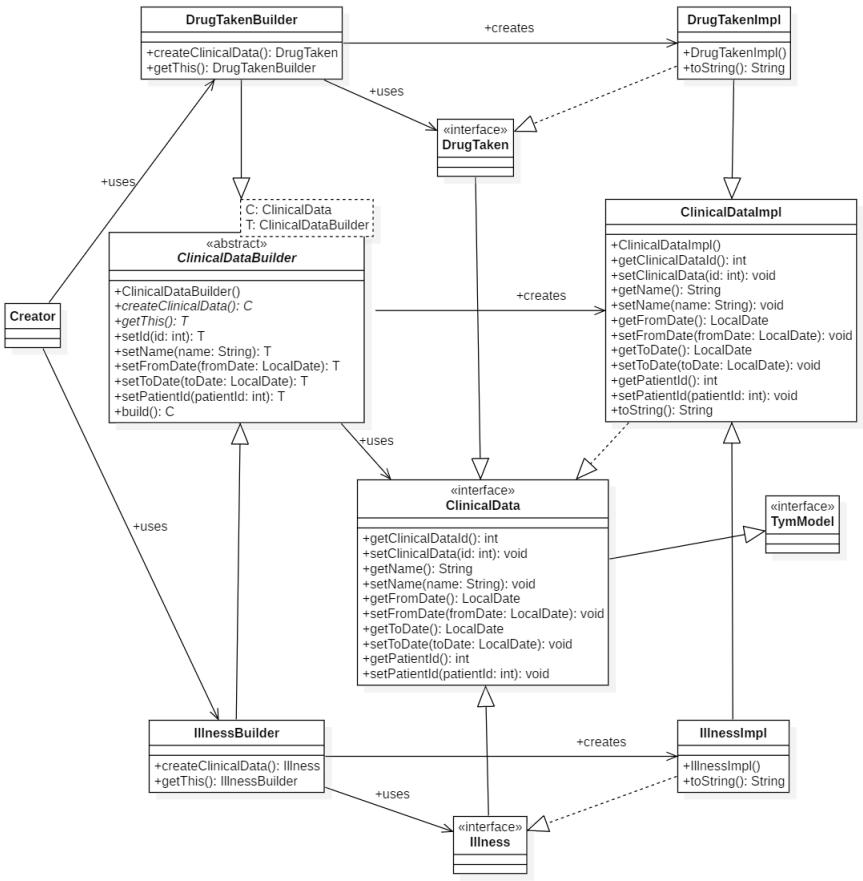


Figura 3.21: Diagramma delle classi del pattern Builder con la gerarchia di classi.

Trattandosi di un'applicazione in JavaFX, tutti i controller devono estendere l'interfaccia `Initializable`, che permette di settare tutti i componenti grafici all'apertura di una vista. Dato che, molti dei controller necessari per gestire le viste prevedono metodi ricorrenti, è stata sviluppata una gerarchia di interfacce, già mostrata in Figura 3.3.

Le viste, invece, sono realizzate in JavaFX FXML, il quale è un formato XML che permette al progettista di comporre la GUI dell'applicazione JavaFX in modo molto simile a una pagina HTML, rendendo più facile la sua manutenzione. Il numero delle viste da creare è pari al numero di schermate ottenute durante la fase di progettazione dell'interfaccia grafica.

Entrando più nel dettaglio per quanto riguarda la progettazione di viste e controller, la struttura delle classi necessarie per l'avvio del programma sono mostrate in Figura 3.22. Qui è possibile vedere come la classe principale `ClientGui` deve impostare lo stage dell'applicazione con le viste iniziali, prima con `LoadingLayout` e, successivamente, con `PatientHomeLayout` e `TopLayout`. Tale classe dovrà estendere la classe astratta `Application`, in quanto si realizza un'applicazione JavaFX, che necessita di metodi specifici per poter avviare il progetto.

I controller definiti per la gestione dei pazienti di un medico e dei nuovi pazienti devono contenere tutti i metodi necessari per richiedere al server lo scaricamento o l'inserimento dei dati nel database. Questi devono, quindi, essere in grado di ottenere il riferimento dell'attore del client, il quale si occuperà di inoltrare le richieste al server. Essi implementeranno, come mostrato in Figura 3.23, l'interfaccia `ControllerTym`, che mette a disposizione i metodi per settare l'attore client e per gestire le eccezioni, quando si verificano.

La definizione dei controller che gestiscono la cartella clinica del paziente è molto simile al caso precedente. Come si può notare dal diagramma di Figura 3.24, `PatientCardTopController`, cioè il controller che gestisce la sezione in alto della scheda del paziente, mette a disposizione degli altri controller una serie di metodi per restituire tutte le informazioni di uno specifico paziente. In questo modo, si eviterà di dover fare troppe chiamate al database quando ci si sposta da una sezione a un'altra.

In alcuni casi, come per il controller che gestisce la vista per assegnare i profili di esercitazione a un paziente (vedi Figura 3.25), deve essere messo a disposizione il metodo che permette di chiudere la scena aperta.

Per gestire i livelli di default assegnati agli esercizi per uno specifico paziente, il controller deve implementare l'interfaccia `LevelControllerTym`, che mette a disposizione dei metodi che restituiscono le liste dei livelli di difficoltà assegnati e degli esercizi presenti nel profilo di esercitazione assegnato al paziente.

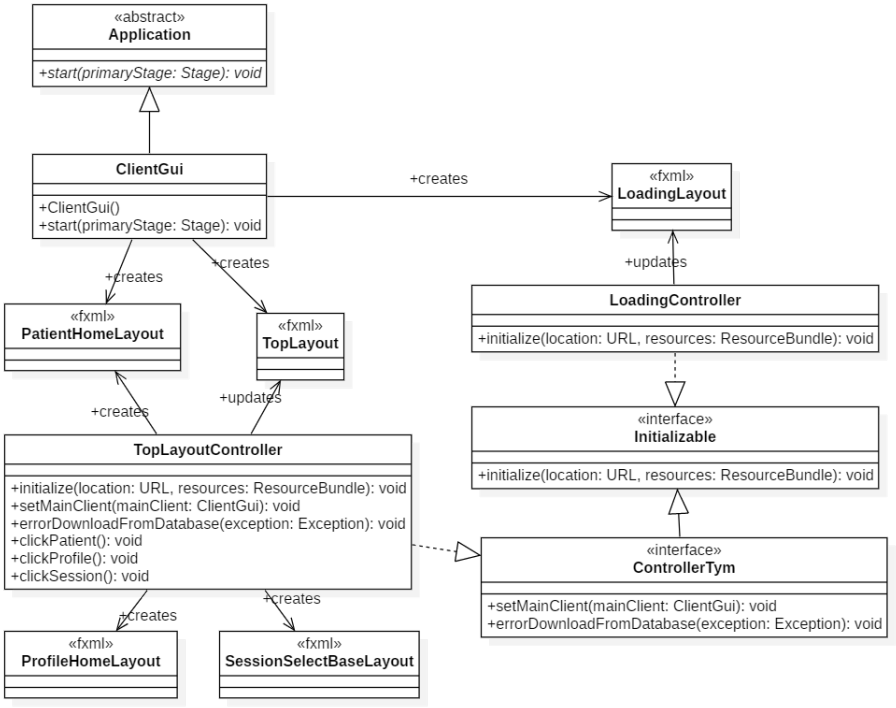


Figura 3.22: Design di dettaglio dell'avvio del programma.

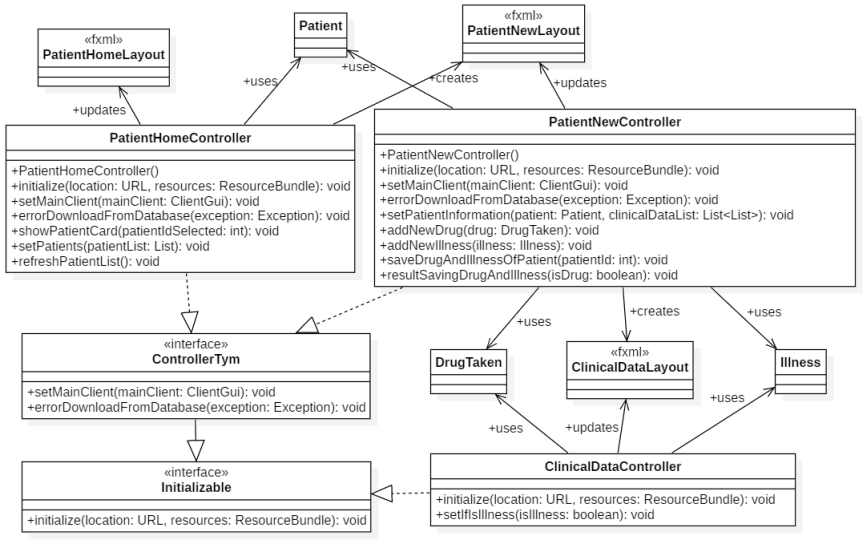


Figura 3.23: Design di dettaglio della gestione dei pazienti ed l'inserimento di un nuovo paziente.

Nella sezione della scheda clinica riguardante le sessioni di esercitazione svolte dal paziente, dovranno essere mostrati anche alcuni grafici, a supporto del medico. Tutti i controller che gestiranno le viste contenenti grafici, avranno la necessità di avere la lista completa degli esercizi svolti dal paziente, perciò essi dovranno implementare l'interfaccia `ChartControllerTym`, come visibile in Figura 3.26.

La sezione riguardante la gestione dei profili di esercitazione, il cui dettaglio è mostrato in Figura 3.27, si basa sullo stesso principio già descritto per la gestione dei pazienti. I controller implementano tutti l'interfaccia `ControllerTym`, perché avranno la necessità di comunicare con il server per poter ottenere le informazioni dei profili e inserire nuovi profili sul database.

La gestione della sessione di training richiede l'utilizzo di controller particolari (vedi Figura 3.28). Innanzitutto, per la navigazione tra le varie scene si è previsto l'utilizzo di una breadcrumb bar, perciò questa dovrà essere gestita con un controller (`BreadcrumbBarController`), che avrà il compito di impostare i singoli livelli della gerarchia di scene e permettere all'utente di tornare indietro.

La scena di selezione del paziente e del profilo di esercitazione sono molto simili, in quanto entrambe contengono la breadcrumb bar e una tabella per selezionare l'elemento di interesse (persona o profilo). Per evitare la duplicazione di codice, si creerà un unico layout gestito da `SessionSelectBaseCon-`

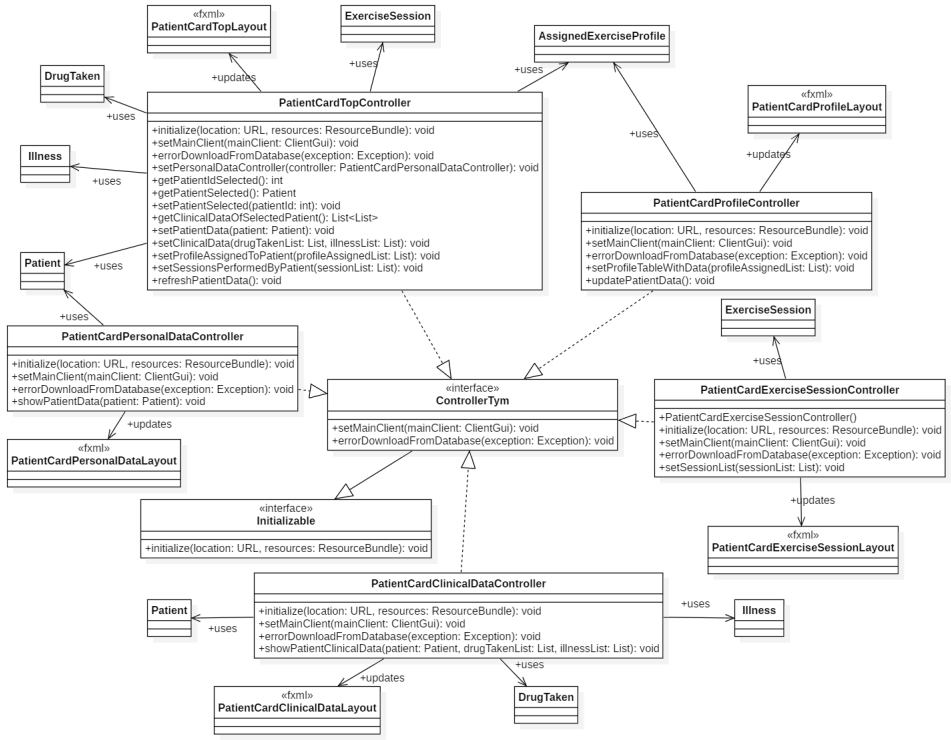


Figura 3.24: Design di dettaglio della scheda clinica di un paziente.

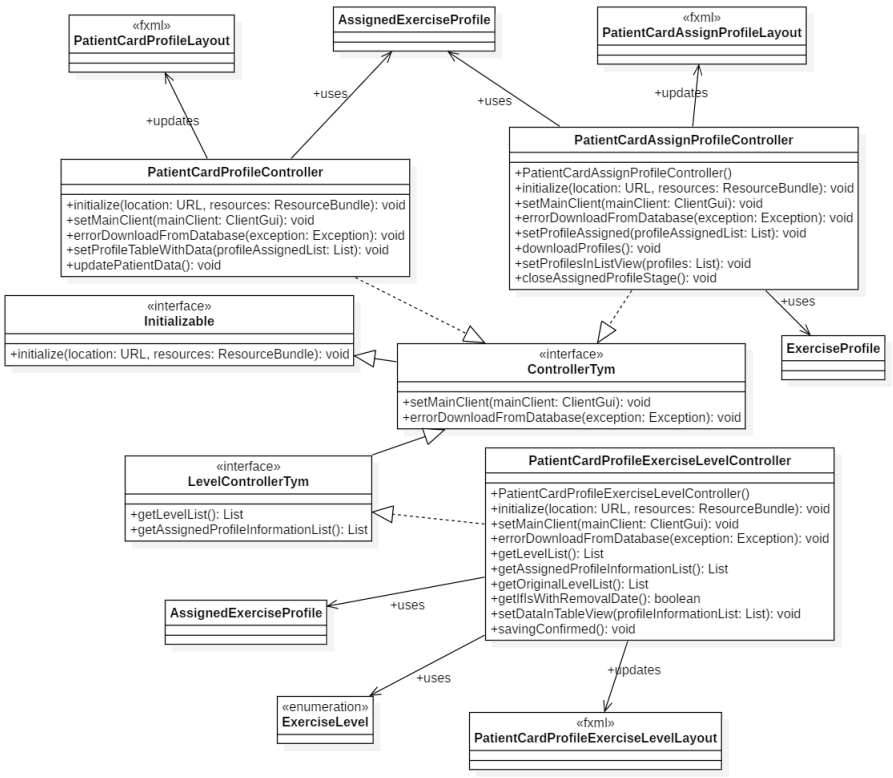


Figura 3.25: Design di dettaglio per la gestione dei profili assegnati a un paziente.

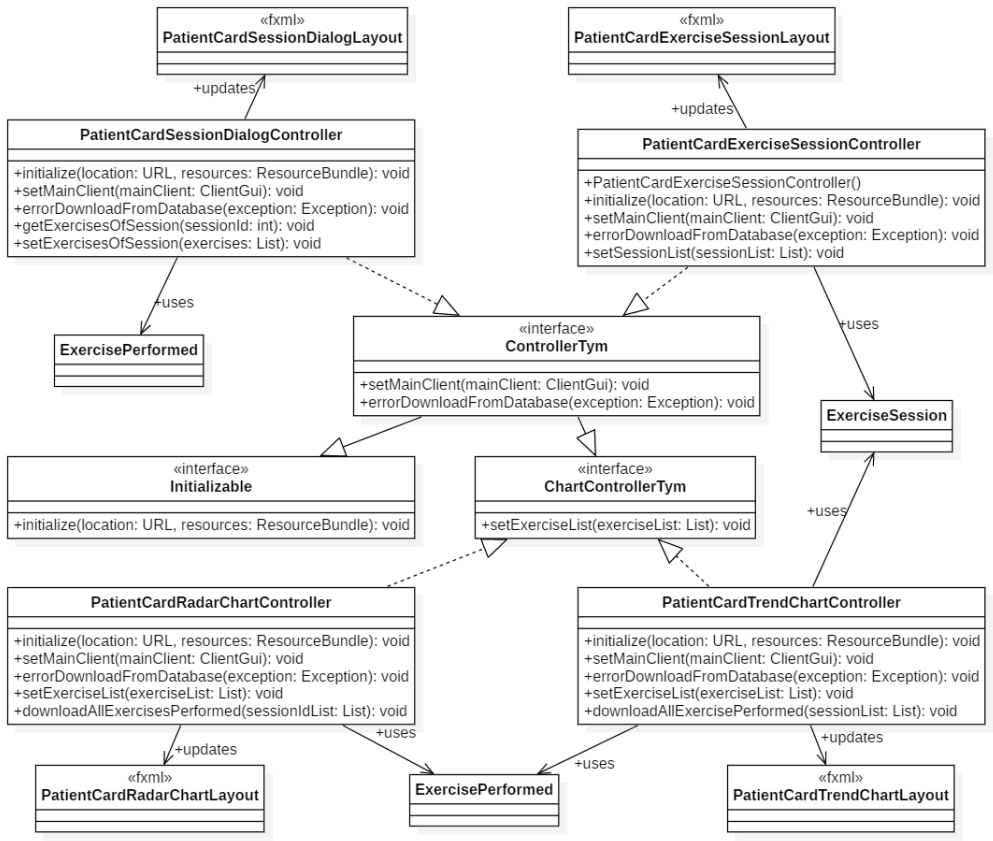


Figura 3.26: Design di dettaglio delle sessioni di esercitazione svolte da un paziente.

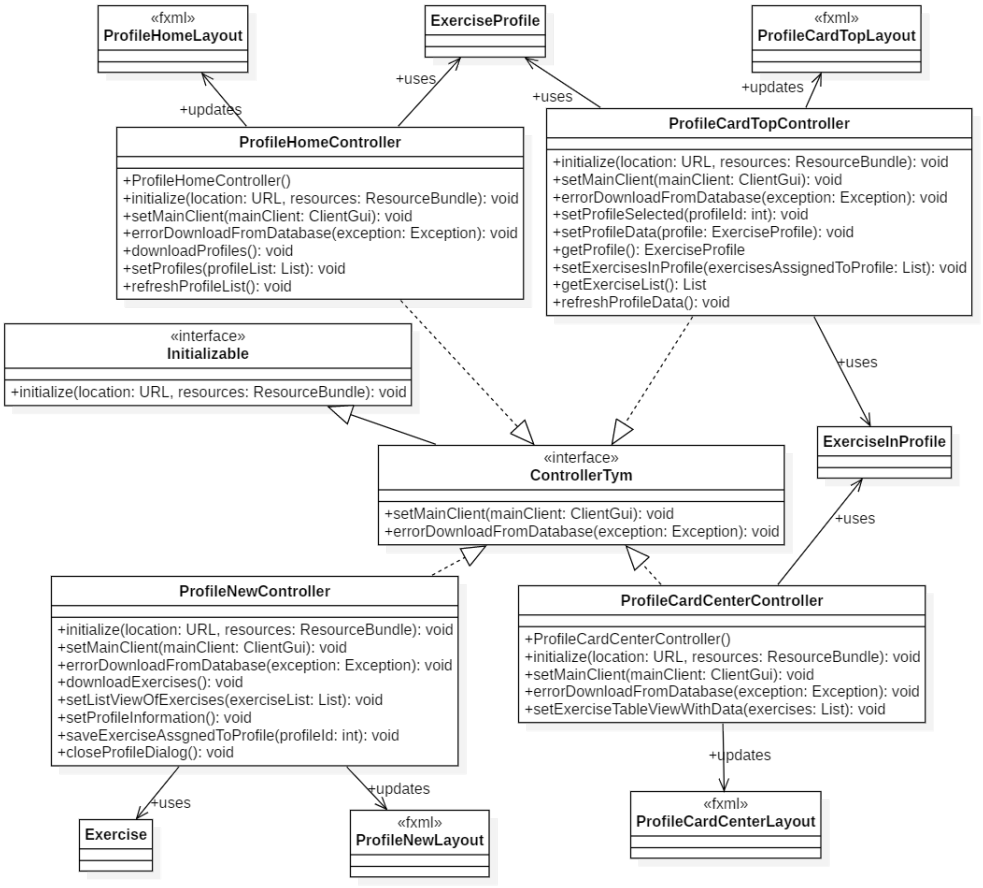


Figura 3.27: Design di dettaglio per la gestione dei profili di esercitazione.

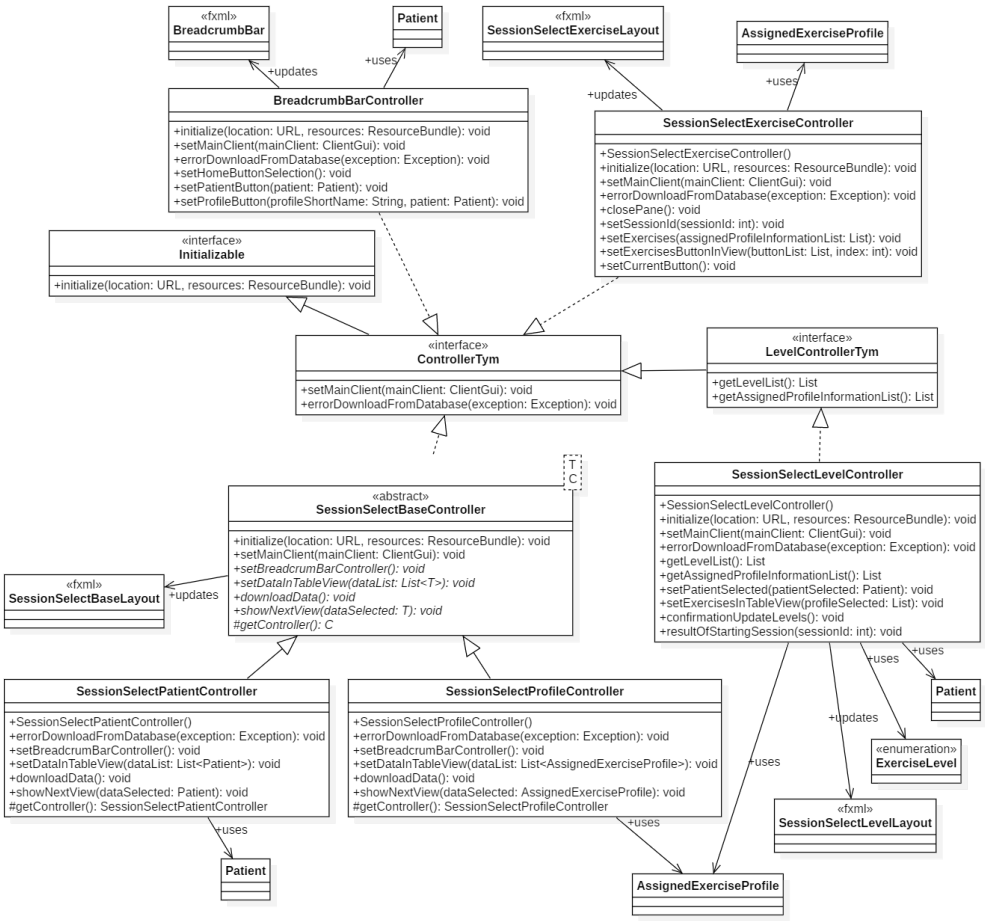


Figura 3.28: Design di dettaglio per l'avvio di una sessione di esercitazione.

troller (JavaFX non permette di assegnare più controller allo stesso layout). Tale controller verrà esteso da `SessionSelectPatientController` e `SessionSelectProfileController`, i quali implementeranno i suoi metodi astratti, permettendo così la personalizzazione del funzionamento.

Infine, i controller per la gestione degli esercizi estendono, invece, l'interfaccia `ExerciseControllerTym`, la quale mette a disposizione metodi per impostare la lista dei componenti grafici contenenti gli esercizi e l'identificativo della sessione di esercitazione che si sta svolgendo, necessario per aggiornare il record della sessione nel database, quando sarà terminata. Il design di questi componenti è visibile in Figura 3.29.

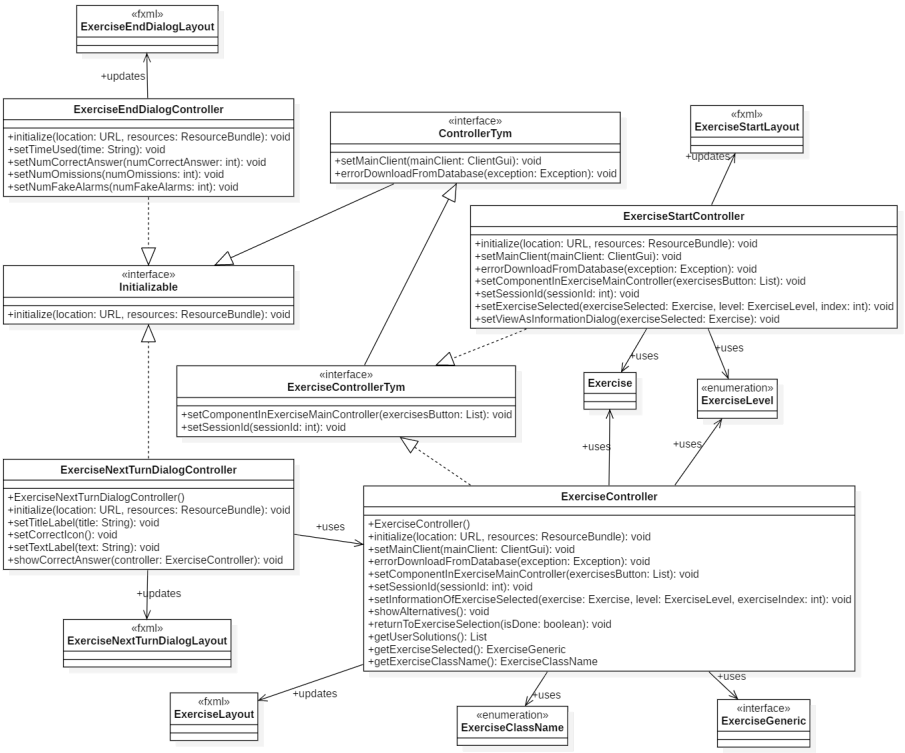


Figura 3.29: Design di dettaglio per la gestione degli esercizi.

3.6 Progettazione del framework degli esercizi

Il fulcro di tutto il progetto è lo sviluppo del framework degli esercizi, che sono necessari per poter svolgere l'attività di allenamento cognitivo sui pazienti.

Un framework è un'architettura logica di supporto su cui si progetta un software. Per poterlo progettare al meglio, è necessario conoscere le principali caratteristiche degli esercizi per trovare peculiarità comuni, che permettano una strutturazione adeguata del codice.

3.6.1 Identificazione degli esercizi

Per poter individuare le caratteristiche che accomunano i vari compiti, è stato necessario, per prima cosa, identificare gli esercizi che sarebbero stati implementati.

Dopo un'attenta riflessione da parte del committente, sono state identificate 6 tipologie di esercizi da realizzare. Due di queste tipologie sono esercizi molto semplici e basilari, mentre tutti gli altri sono i tipici esercizi che vengono svolti nelle tradizionali sedute di training.

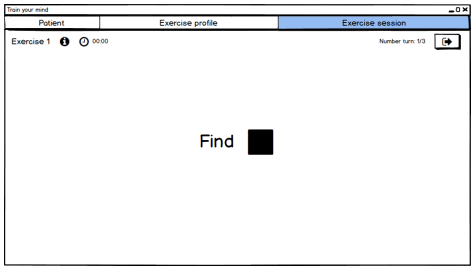
Qui di seguito si descrivono gli esercizi da implementare, indicandone il tipo di compito, le caratteristiche salienti e il loro tipico funzionamento.

Matrici attentive Compito di attenzione selettiva, dove vengono presentati una serie di stimoli di vario genere all'interno di un riquadro. L'obiettivo del paziente è quello di individuare, nel più breve tempo possibile tutti gli stimoli presenti di un certo tipo.

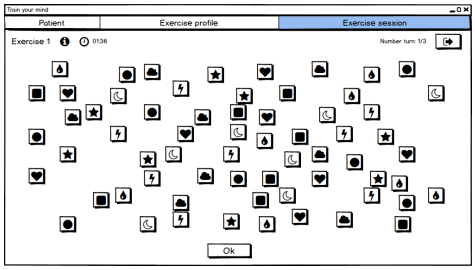
Uno stimolo è una figura che può essere geometrica, come ad esempio un quadrato, un cerchio, un triangolo o un rombo, oppure un simbolo, come ad esempio un cuore, una saetta o un asterisco.

All'avvio dell'esercizio viene richiesto al paziente di memorizzare quale tipo di stimolo deve ricercare, poi gli viene sottoposta una matrice di simboli, in cui ricercare lo stimolo. Al termine dell'esercizio si va a valutare il tempo impiegato, il numero di risposte corrette date, il numero di omissioni e il numero di falsi allarmi. In ogni turno dell'esercizio, si sottopone al paziente la medesima matrice di stimoli.

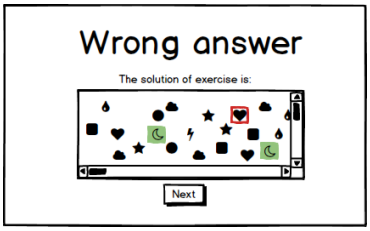
Questo esercizio si svolge in quattro diverse tipologie, dove in ogni tipologia si varia il numero di tipi di stimoli presentati. Le tipologie sono: a 8, a 10, a 11 e a 13 tipi di stimoli. Inoltre, ogni tipologia può essere svolta in tre diversi livelli di difficoltà, i quali variano in base al numero di figure che compaiono per ogni singolo stimolo. I livelli di difficoltà sono:



(a) Stimolo da memorizzare e cercare.



(b) Matrice con gli stimoli.



(c) Dialog con la soluzione corretta.

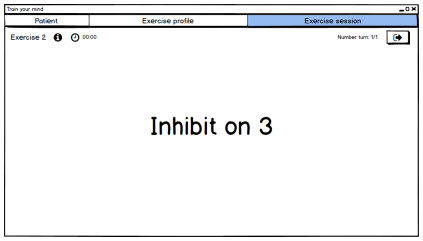
Figura 3.30: Mockup per l'esercizio sulle matrici attentive.

- il livello principiante con 8 figure per stimolo;
- il livello intermedio con 16 figure per stimolo;
- il livello avanzato con 24 figure per stimolo.

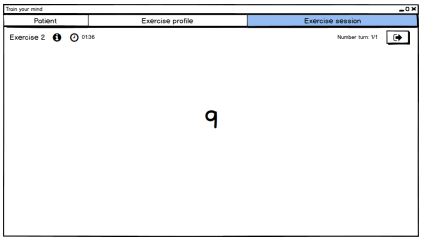
Per la sua trasposizione in versione computerizzata si prevede di mostrare una matrice, come quella di Figura 3.30b, in cui verranno inseriti gli stimoli in modo random e posizionati all'interno della cella della matrice in modo diverso (si spostano in alto, in basso, a destra, a sinistra o al centro). L'utente dovrà selezionare tutte le figure uguali allo stimolo memorizzato (Figura 3.30a) e una volta terminata la selezione deve confermarla per procedere alla verifica della soluzione. Il numero totale di ripetizioni da effettuare in un esercizio sulla medesima matrice sono tre.

Nel caso in cui il paziente individui una soluzione errata, dovrà essere mostrata la stessa matrice su cui si è svolto l'esercizio e vi dovranno essere indicate le posizioni di ciascuno stimolo corretto, come mostrato in Figura 3.30c.

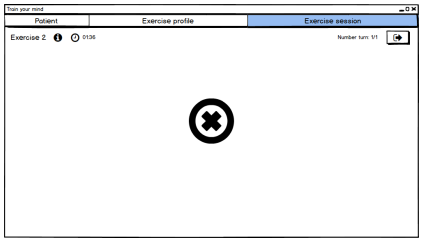
SART Compito di attenzione sostenuta, dove vengono mostrati a schermo una sequenza di numeri (Figura 3.31b), tra 1 e 9, alternati da un simbolo (Figura 3.31c). L'obiettivo dell'utente è quello di premere la barra spaziatrice



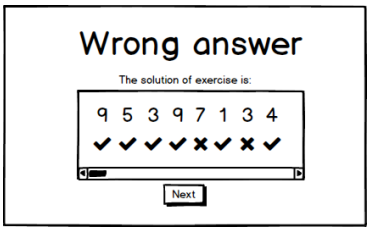
(a) Stimolo su cui inibirsi.



(b) Sequenza di numeri.



(c) Simbolo di intervallo.



(d) Dialog con la soluzione corretta.

Figura 3.31: Mockup dell'esercizio SART.

alla comparsa di ogni numero, ma di inibirsi quando compare uno specifico numero definito a priori.

Il paziente deve, quindi, memorizzare il numero su cui si deve inibire (Figura 3.31a), poi gli viene fatto svolgere l'esercizio, il quale ha una durata di 10 minuti.

Per questo esercizio si valuta il tempo di risposta dell'utente e l'accuratezza (in base al numero di risposte corrette, omissioni e falsi allarmi).

Il funzionamento dell'esercizio è del tutto temporizzato, infatti ogni numero è visualizzato per 250 millisecondi, mentre il simbolo che intervalla i numeri ha una durata che varia in base al livello di difficoltà selezionato:

- nel livello principiante la durata è di 2000 millisecondi;
- nel livello intermedio la durata è di 1500 millisecondi;
- nel livello avanzato la durata è di 1000 millisecondi.

Di questo compito esisteva già una versione computerizzata, ma non prevedeva la generazione random del numero su cui ci si deve inibire, obbligando così il paziente a inibirsi sempre e solo sul numero 3.

In caso di risposta errata, dovrà essere mostrata la sequenza di numeri proposta e per ogni numero si dovrà indicare se la risposta data dal paziente era corretta o errata, come mostrato in Figura 3.31d.

Attenzione alternata Compito di attenzione complessa, dove sono mostrati a schermo cinque figure geometriche (Figura 3.32b) con all'interno un numero (compreso tra 1 e 5). L'obiettivo dell'utente è quello di indicare il numero presente all'interno di una specifica figura geometrica, che viene decisa a priori. La figura geometrica da cercare non è sempre uguale per tutto l'esercizio, ma va alternata con un'altra alla comparsa di un simbolo di cambio (come ad esempio un asterisco oppure come quello di Figura 3.32c).

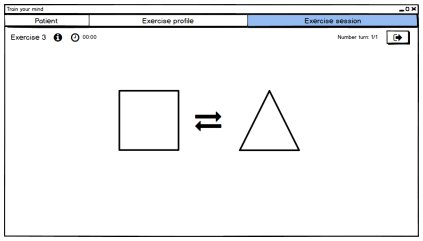
Per comprendere meglio il funzionamento di tale esercizio, verrà fatto un esempio pratico. Si immagini di dover cercare i numeri prima all'interno del quadrato e poi all'interno del cerchio. Una volta avviato l'esercizio, il paziente inizierà ad indicare i numeri presenti all'interno del quadrato. Ad un certo punto, verrà mostrato il simbolo di cambio, allora il paziente inizierà ad indicare i numeri presenti all'interno del cerchio. Alla successiva comparsa del simbolo di cambio, il paziente tornerà ai numeri all'interno del quadrato. Il paziente procederà in questo modo, alternandosi tra le due figure, fino al termine dell'esercizio, dove avrà individuato 20 numeri.

Come nell'esercizio descritto precedentemente, anche questo ha un funzionamento temporizzato. La schermata con le cinque figure geometriche rimane a schermo fino a quando non viene premuta la barra spaziatrice, mentre la schermata bianca, che separa due schermate con le figure geometriche, e la schermata del simbolo di cambio hanno una durata di 1 secondo.

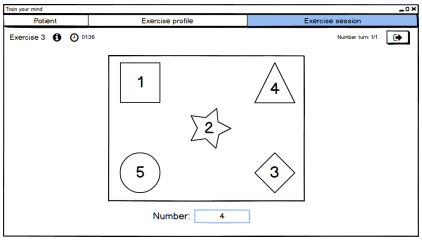
La scelta del livello fa variare il numero di volte in cui si deve cercare il numero all'interno di una figura, prima di passare a quella successiva. Le possibilità a disposizione sono:

- nel livello principiante si deve cercare la figura 5 volte prima di vedere il simbolo di cambio;
- nel livello intermedio si deve cercare la figura 3 volte prima del simbolo di cambio;
- nel livello avanzato si deve cercare la figura 1 volta prima del simbolo di cambio,
- nel livello randomizzato si deve cercare la figura ogni 5, 3 o 1 volta, in base del numero generato in modo casuale, prima di vedere il simbolo di cambio.

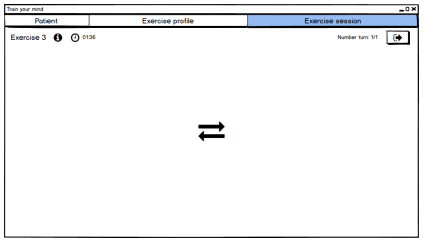
Anche in questo caso, esiste già una sua versione computerizzata, ma si ha un grosso problema, in quanto non permette la completa autonomia di esecuzione da parte del paziente. Infatti, in quella versione, era il medico a decidere le due figure tra cui alternarsi e il paziente doveva dire a voce



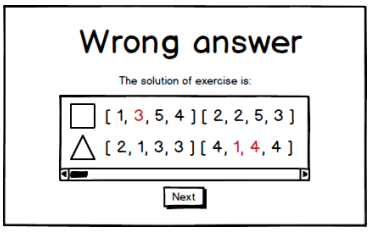
(a) Figure da alternare.



(b) Configurazione delle figure.



(c) Simbolo di cambio.



(d) Dialog con la soluzione corretta.

Figura 3.32: Mockup per l'esercizio sull'attenzione alternata.

il numero all'interno della figura, per poi premere la barra spaziatrice per ottenere la configurazione seguente. Era compito del medico verificare se la risposta data dal paziente era corretta o meno.

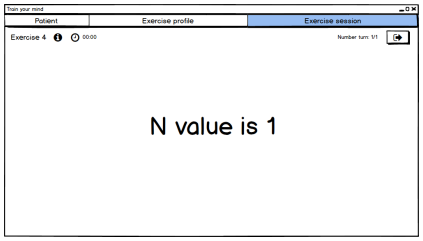
Nella versione all'interno di "Train your mind", le figure tra cui alternarsi sono mostrate al paziente poco prima di cominciare l'esercizio (vedi Figura 3.32a). Inoltre, durante lo svolgimento dell'esercizio, il paziente deve indicare il numero all'interno della figura, digitandolo in un campo apposito. A seguito della digitazione si passa direttamente alla configurazione seguente, senza bisogno di dover premere altri tasti.

Al termine dell'esercizio, in caso di soluzione errata del paziente, verrà mostrata, per ogni figura da alternare, i numeri corretti, come in Figura 3.32d.

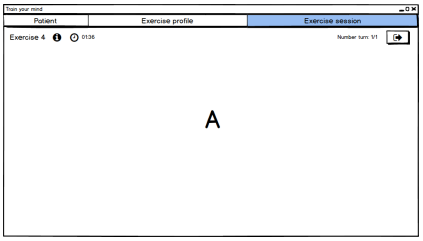
N-back Compito che allena la memoria lavoro, in cui viene mostrata una sequenza di numeri (da 1 a 9), alternati da una schermata vuota. L'obiettivo del paziente è quello di confrontare il numero attuale con quello mostrato N posizioni prima e, se sono uguali, di premere la barra spaziatrice. Al termine dell'esercizio si valuta l'accuratezza delle risposte date.

La durata totale dell'esercizio è di 3 minuti, durante i quali i numeri sono mostrati per 1 secondo, mentre la schermata vuota per 2 secondi.

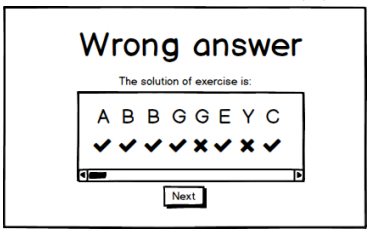
La selezione del livello di difficoltà fa variare, invece, il valore di N:



(a) Valore di N.



(b) Sequenza di lettere.



(c) Dialog con la soluzione corretta.

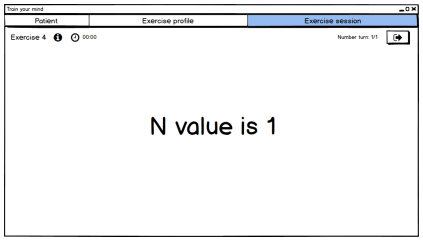
Figura 3.33: Mockup dell'esercizio N-back.

- nel livello principiante il valore di N è pari a 1, perciò l'utente dovrà confrontare il numero corrente con quello mostrato prima;
- nel livello intermedio il valore di N è pari a 2, perciò l'utente confronta il numero corrente con quello mostrato due posizioni prima.

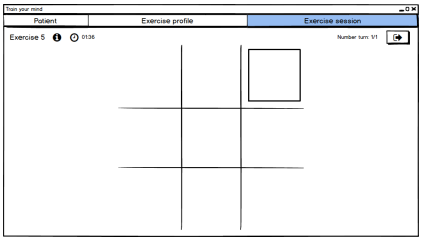
La versione dell'esercizio appena descritta è quella utilizzata nei test di valutazione cognitiva tradizionali, perciò per fare in modo che l'allenamento svolto sia utile a migliorare anche il risultato nel test di valutazione ed evitare che il paziente si abitui al tipo di compito da svolgere, si è deciso di sviluppare una versione in cui si mostra una sequenza di lettere dell'alfabeto (Figura 3.33b). È importante fare in modo che le due lettere uguali compaiano in una quantità pari al 15% rispetto al numero totale degli stimoli mostrati, cosa che si verificava con molta più facilità quando si trattava di generare numeri random a una sola cifra, rispetto alle 26 lettere.

In caso di soluzione errata del paziente, si dovrà mostrare la sequenza di lettere proposta, indicando dove sono stati commessi gli errori, come mostrato in Figura 3.33c.

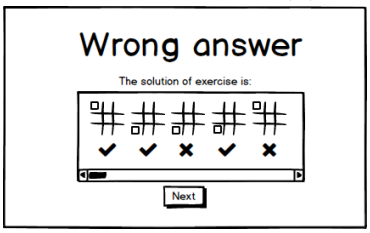
La seconda versione di questo compito, che si può chiamare *Position N-back* (Figura 3.34), vede l'alternarsi della posizione di un quadrato all'interno di una griglia 3×3. Il funzionamento è identico al caso precedente, ma l'utente dovrà premere la barra spaziatrice quando il quadrato corrente è nella stessa posizione del quadrato mostrato N posizioni prima.



(a) Valore di N.



(b) Sequenza di quadrati.



(c) Dialog con la soluzione corretta.

Figura 3.34: Mockup dell'esercizio Position N-back.

Stop signal Compito di attenzione sostenuta in cui viene mostrato un quadrato che può cambiare colore. L'obiettivo del paziente è quello di premere la barra spaziatrice quando la figura è di colore rosso (Figura 3.35b), mentre si deve inibire quando è di colore grigio (Figura 3.35a) oppure quando il quadrato è sormontato da un arco rosso (Figura 3.35c).

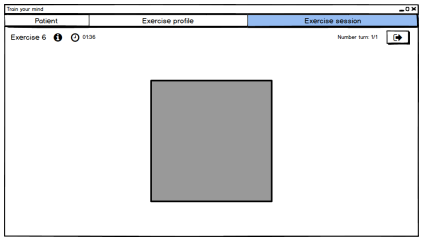
La durata totale dell'esercizio è di 3 minuti, durante i quali la schermata con il quadrato ha una durata di 2 secondi, mentre la schermata vuota di 1 secondo.

Il livello di difficoltà va a variare il tempo che impiega l'arco rosso a comparire sopra il quadrato rosso:

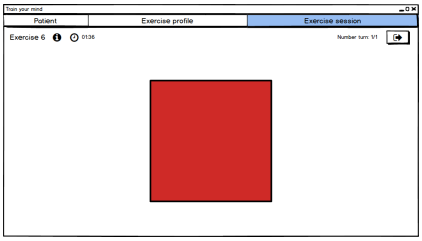
- nel livello principiante compare nello stesso momento del quadrato;
- nel livello intermedio compare dopo 500 millisecondi;
- nel livello avanzato compare dopo 1 secondo.

Nel caso in cui il paziente sottometta una soluzione errata, si dovrà, come anche negli esercizi già descritti, mostrare la sequenza di figure proposte e per ognuna di esse indicare gli errori commessi dall'utente, come in Figura 3.35d.

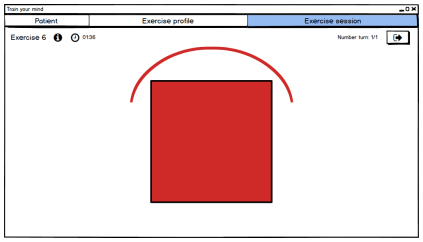
Al termine dell'esercizio si valuta l'accuratezza delle risposte date dal paziente, in base al numero di risposte corrette, numero di omissioni e numero di falsi allarmi.



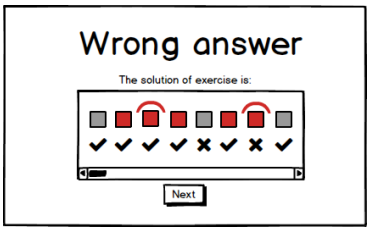
(a) Quadrato grigio.



(b) Quadrato rosso.



(c) Quadrato rosso con l'arco.



(d) Dialog con la soluzione corretta.

Figura 3.35: Mockup dell'esercizio Stop signal.

Ordinamento crescente Compito di attenzione complessa, in cui sono presentati dei numeri (come in Figura 3.36a). L'obiettivo dell'utente è quello di ordinare i numeri in ordine crescente (dal più piccolo al più grande), selezionando i numeri secondo il giusto ordine.

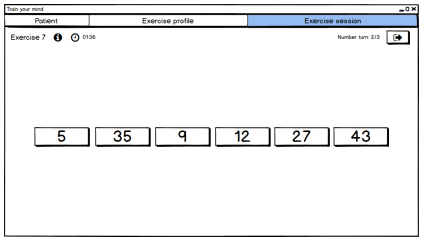
In questo esercizio, si valutano la rapidità e la capacità del paziente di elaborare le informazioni presentate e restituire la giusta soluzione al problema.

Il range entro il quale si generano i numeri proposti varia in base al livello di difficoltà con cui viene eseguito l'esercizio:

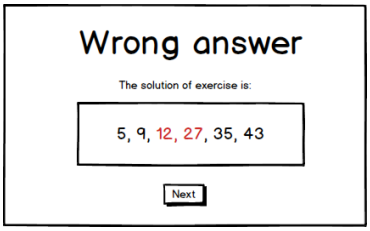
- nel livello principiante si generano numeri tra 1 e 49;
- nel livello intermedio si generano numeri tra 1 e 99;
- nel livello avanzato si generano numeri tra 1 e 299.

Come sempre, in caso di risposta errata da parte del paziente, si dovrà mostrare la soluzione corretta che doveva essere trovata (Figura 3.36b).

Combinare la figura al dettaglio Compito di tipo percettivo motorio, dove viene presentato il dettaglio di una figura. L'obiettivo del paziente è quello di indicare a quale delle tre figure proposte appartiene il dettaglio mostrato, come visibile in Figura 3.37a.

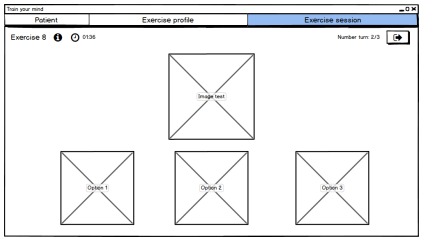


(a) Numeri da ordinare.

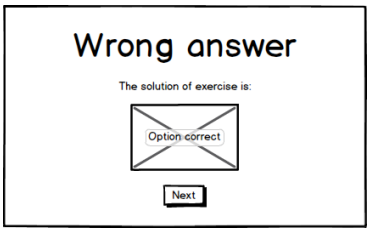


(b) Dialog con la soluzione corretta.

Figura 3.36: Mockup per l'esercizio dell'ordinamento crescente.



(a) Dettaglio e figure proposte.



(b) Dialog con la soluzione corretta.

Figura 3.37: Mockup dell'esercizio di combinare la figura al dettaglio.

La scelta del livello di difficoltà da avviare fa variare il numero di parti in cui l'immagine viene scomposta per ottenere il dettaglio:

- nel livello principiante l'immagine è scomposta in 4 parti;
- nel livello intermedio l'immagine è scomposta in 25 parti;
- nel livello avanzato l'immagine è scomposta in 100 parti.

Al termine dell'esercizio, si deve verificare se la risposta data è quella corretta, altrimenti andrà mostrata la figura a cui il dettaglio apparteneva (Figura 3.37b).

3.6.2 Progettazione delle classi e del funzionamento degli esercizi

Lo sviluppo del framework richiede che vengano progettate delle classi che possano essere estese per qualsiasi tipologia di compito si debba realizzare. In questo modo, qualunque sviluppatore, partendo da una base comune, potrà

implementare il suo personale esercizio, senza dover andare a modificare quelli già presenti nel progetto.

Perciò, una volta analizzati gli esercizi richiesti, è stato possibile individuare le seguenti caratteristiche comuni:

- tutti i compiti hanno una serie di alternative che vengono proposte all'utente. Può trattarsi dell'alternanza di un testo o figure mostrate a video oppure di opzioni tra cui scegliere per dare la soluzione;
- tutti i compiti hanno una soluzione finale, la quale permetterà, attraverso un confronto, di affermare se la risposta del paziente è corretta;
- tutti i compiti necessitano di una funzione che restituisca la correzione della soluzione dell'utente. La funzione restituirà il numero di risposte corrette ed errate oppure il numero di risposte corrette, il numero di omissioni (l'utente non ha risposto quando avrebbe dovuto) e il numero di falsi allarmi (l'utente ha risposto quando non avrebbe dovuto);
- alcuni esercizi hanno anche un test, cioè qualcosa che può dover essere memorizzato e che va confrontato con le alternative mostrate durante il corso dell'esercizio. Il test può dover essere mostrato durante tutto lo svolgimento del compito, come ad esempio nel caso del dettaglio di una figura, oppure può essere mostrato per un breve periodo di tempo e poi fatto scomparire, come ad esempio lo stimolo nelle matrici attentive oppure il numero su cui ci si deve inibire nel SART;
- alcuni compiti hanno la necessità di mostrare le alternative solo per un breve lasso di tempo, perciò per ogni alternativa si dovrà indicare la quantità di tempo entro la quale sarà mostrata a schermo.

La scoperta di queste proprietà che accomunano i singoli compiti ha permesso di progettare le seguenti tre interfacce, mostrate nel diagramma di Figura 3.38:

- `ExerciseGeneric`, che modella un semplice esercizio di base, dove si hanno delle opzioni tra cui l'utente deve scegliere, una soluzione corretta e una funzione per controllare la soluzione dell'utente.
- `ExerciseGenericWithTest`, che estende `ExerciseGeneric`, a cui aggiunge un metodo per modellare anche il test da mostrare all'utente.
- `ExerciseGenericWithLapse`, che modella un esercizio con le alternative temporizzate.

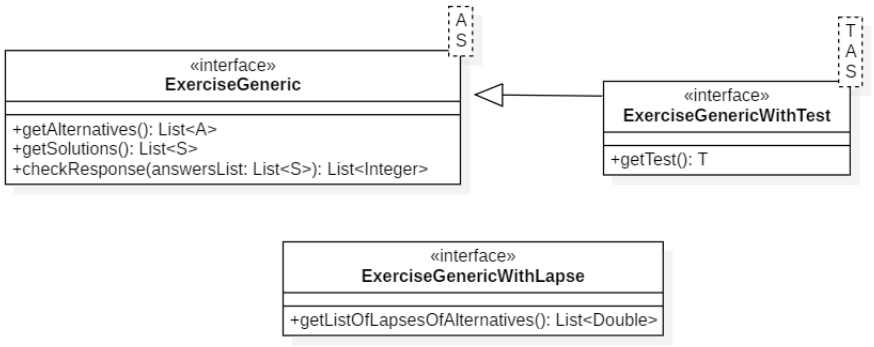


Figura 3.38: Diagramma delle classi che mostra le tre interfacce base del framework degli esercizi.

Per il progetto corrente, le interfacce sopra descritte saranno implementate dalle classi dei compiti identificati. Ogni classe implementerà tutte o parte delle interfacce, a seconda dei metodi che sono necessari per il suo corretto funzionamento. Qui di seguito si presenta l'elenco delle classi progettate, con le loro principali caratteristiche e corredate da un diagramma delle classi per capirne meglio la struttura:

- `AttentionalMatrix`, con le sue quattro estensioni `AttentionalMatrix8Symbols`, `AttentionalMatrix10Symbols`, `AttentionalMatrix11Symbols` e `AttentionalMatrix13Symbols`, implementa l'esercizio delle matrici attentive. Come si può vedere da Figura 3.39, il compito necessita di un test con lo stimolo da cercare, di una lista di alternative e una lista di soluzioni corrette, perciò va ad implementare l'interfaccia `ExerciseGenericWithTest`.

- `Sart`, il quale, a differenza del compito precedente, oltre ad aver bisogno di un test con il numero su cui inibirsi, di una lista di alternative e di una lista di soluzioni corrette, necessita anche di una lista con la durata delle singole alternative. Per questo motivo, oltre ad implementare `ExerciseGenericWithTest`, implementa anche `ExerciseGenericWithLapse` (Figura 3.40).

- `AlternatingAttention`, che modella l'esercizio di attenzione alternata, ha le stesse necessità del compito precedente e perciò implementa `ExerciseGenericWithTest` e `ExerciseGenericWithLapse` (vedi Figura 3.41).

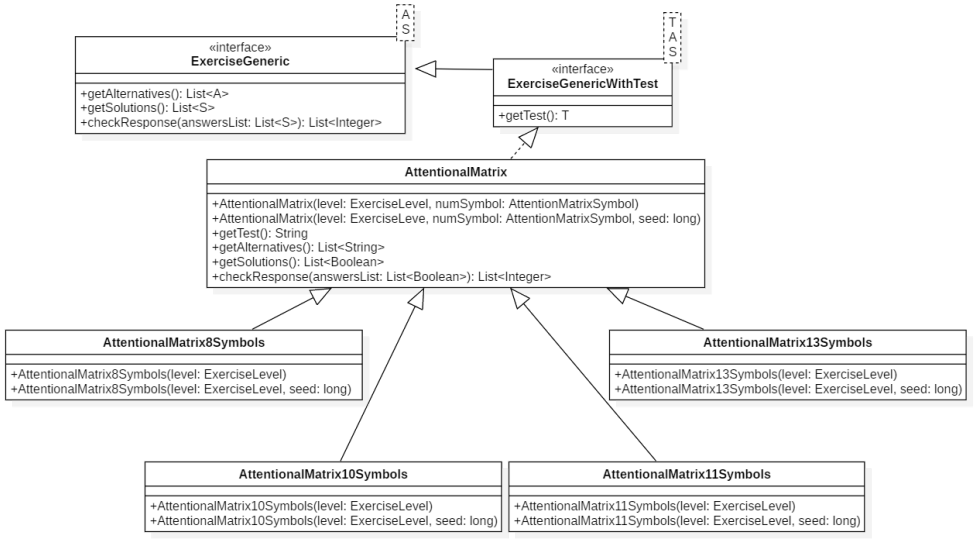


Figura 3.39: Diagramma delle classi per l'esercizio sulle matrici attentive.

- `NBack`, a differenza dei precedenti, è un caso particolare. Come negli altri casi implementa sempre `ExerciseGenericWithTest` e `ExerciseGenericWithLapse`, ma la classe sarà astratta, perché la tipologia delle alternative da presentare varia a seconda se si tratta di lettere (`LetternNBack`) o quadrati (`PositionNBack`). La sua struttura è visibile in Figura 3.42.

- `StopSignal` è del tutto simile alle classi precedenti, ma non necessita di un test da mostrare all'utente. Questo significa che le due interfacce che implementa sono `ExerciseGeneric` e `ExerciseGenericWithLapse` (Figura 3.43).

- `AscendingOrder`, compito molto più basilare dei precedenti, il quale necessita solamente di alternative e una lista di soluzioni con cui confrontare la risposta dell'utente. Come mostrato in Figura 3.44, implementa solamente l'interfaccia `ExerciseGeneric`.

- `CombineFigureToDetail`, il quale necessita, oltre alle alternative, anche di un test da mostrare al paziente. L'interfaccia implementata, vedi Figura 3.45, è `ExerciseGenericWithTest`.

Il funzionamento interno degli esercizi, che è stato progettato, è unico per tutti gli esercizi del framework. Per far variare il funzionamento in base alle

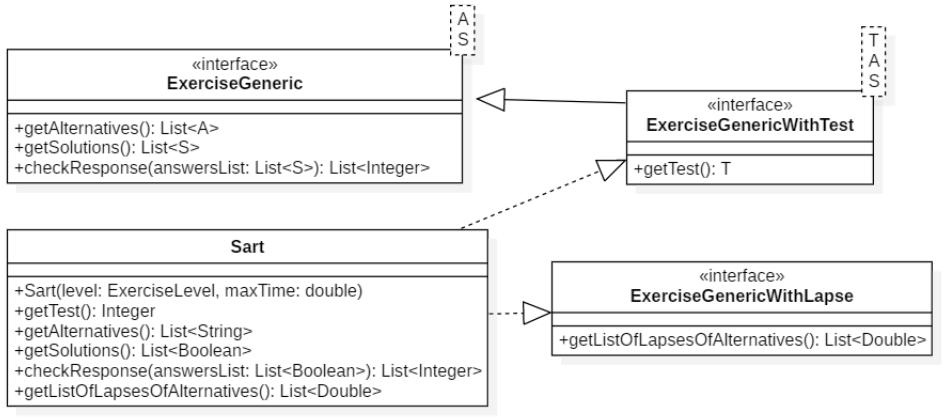


Figura 3.40: Diagramma delle classi per l'esercizio SART.

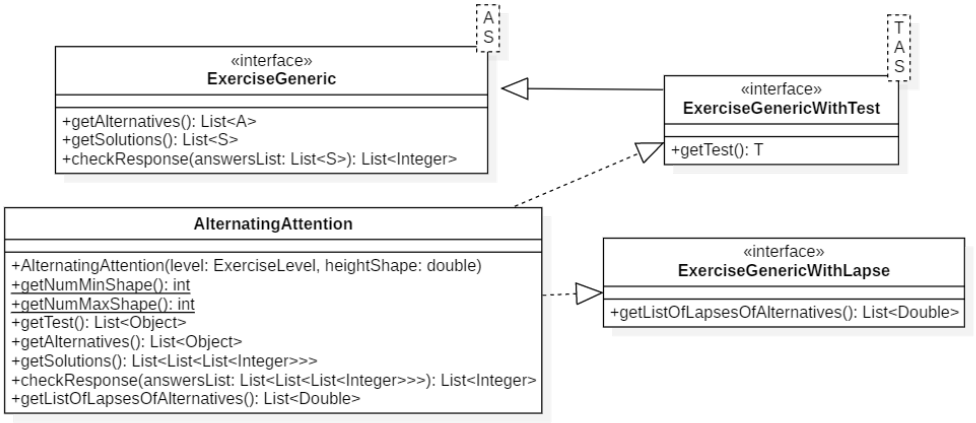


Figura 3.41: Diagramma delle classi per l'esercizio di attenzione alternata.

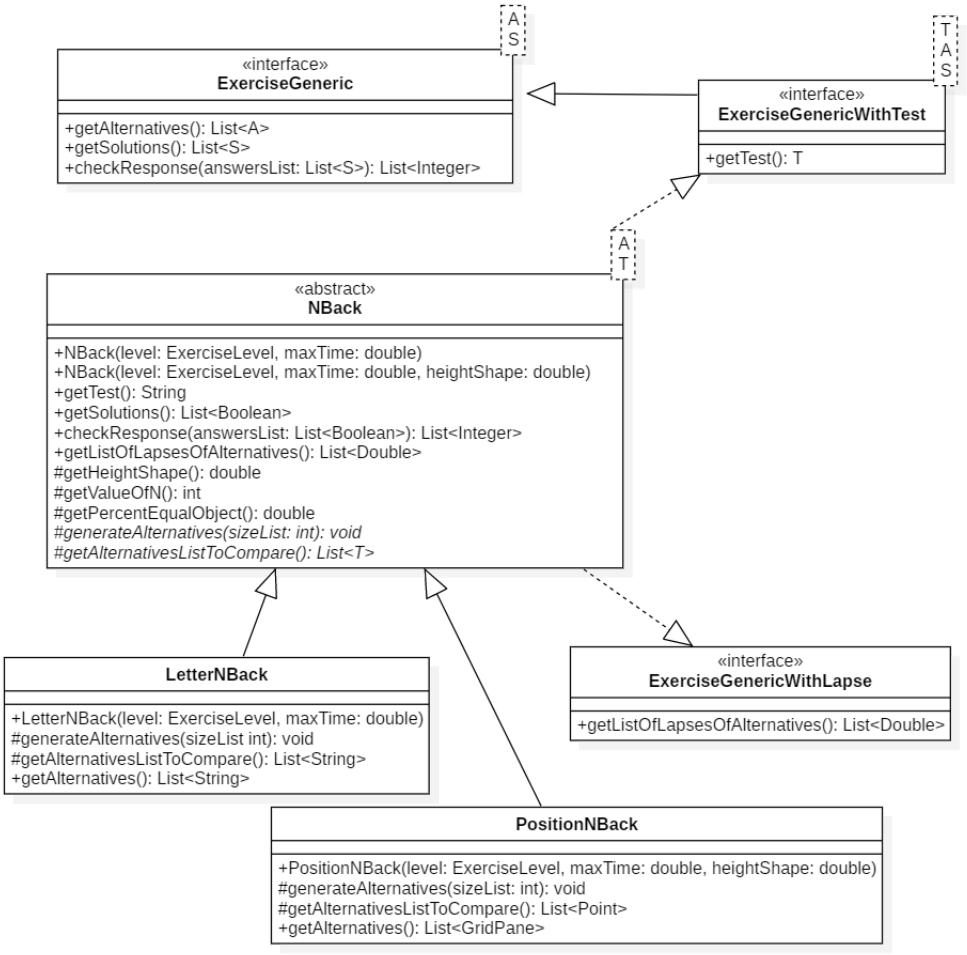


Figura 3.42: Diagramma delle classi per gli esercizi di N-back.

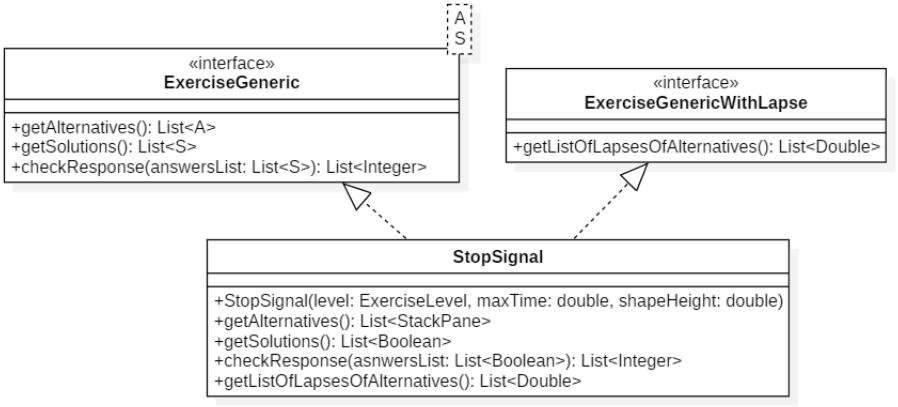


Figura 3.43: Diagramma delle classi per l'esercizio Stop signal.

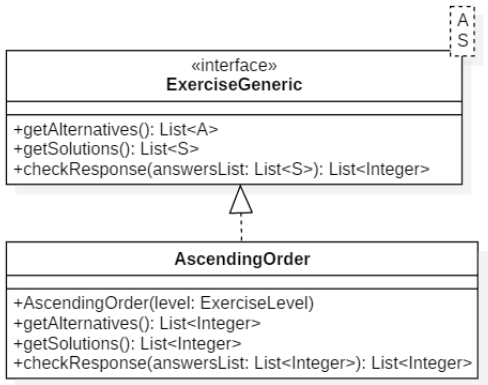


Figura 3.44: Diagramma delle classi per l'esercizio di ordinamento crescente.

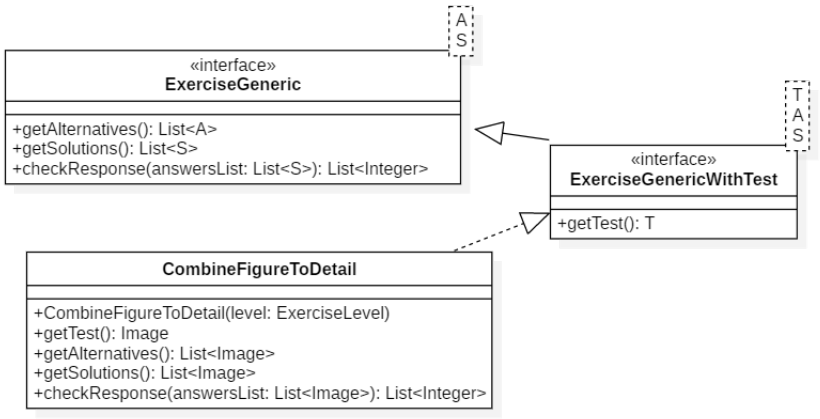


Figura 3.45: Diagramma delle classi per l'esercizio per combinare la figura al dettaglio.

necessità del singolo esercizio sono stati introdotti dei controlli. Il diagramma di Figura 3.46 mostra tutte le azioni da svolgere per ottenere il corretto funzionamento di un esercizio.

All'avvio, si deve creare l'esercizio (la tipologia dipenderà poi dalla selezione fatta dall'utente in una fase precedente). Una volta creato l'esercizio, si deve recuperare il test, se presente per il compito, e lo si mostra sempre o per un tempo limitato. Dopo aver recuperato il test, oppure se l'esercizio non aveva il test, si devono recuperare tutte le alternative del compito e si fa partire il tempo. Le alternative sono estratte, una alla volta, da una lista e, per ognuna di esse, si può impostare il tempo di visibilità, se necessario. Al termine della generazione di tutte le alternative, se non si attende una qualche azione dell'utente per poter procedere, si va a verificare la soluzione ottenuta dall'utente. Nel caso in cui, invece, si attente l'azione dell'utente, la verifica della soluzione sarà fatta solo a seguito dello svolgimento dell'operazione. Durante la verifica della soluzione, il tempo viene fermato, poi si procede con una serie di controlli. Il primo riguarda il numero del turno, perché se si è svolto l'ultimo si deve fermare il tempo e salvare il valore del tempo impiegato per svolgere l'intero compito. Una volta mostrate le dialog per la risposta corretta o sbagliata, se si è giunti al termine dell'esercizio, si mostra la schermata di termine e si salva il risultato dell'esercizio. In caso contrario, si incrementa il valore del turno e si riparte con lo svolgimento dell'esercizio per il nuovo turno.

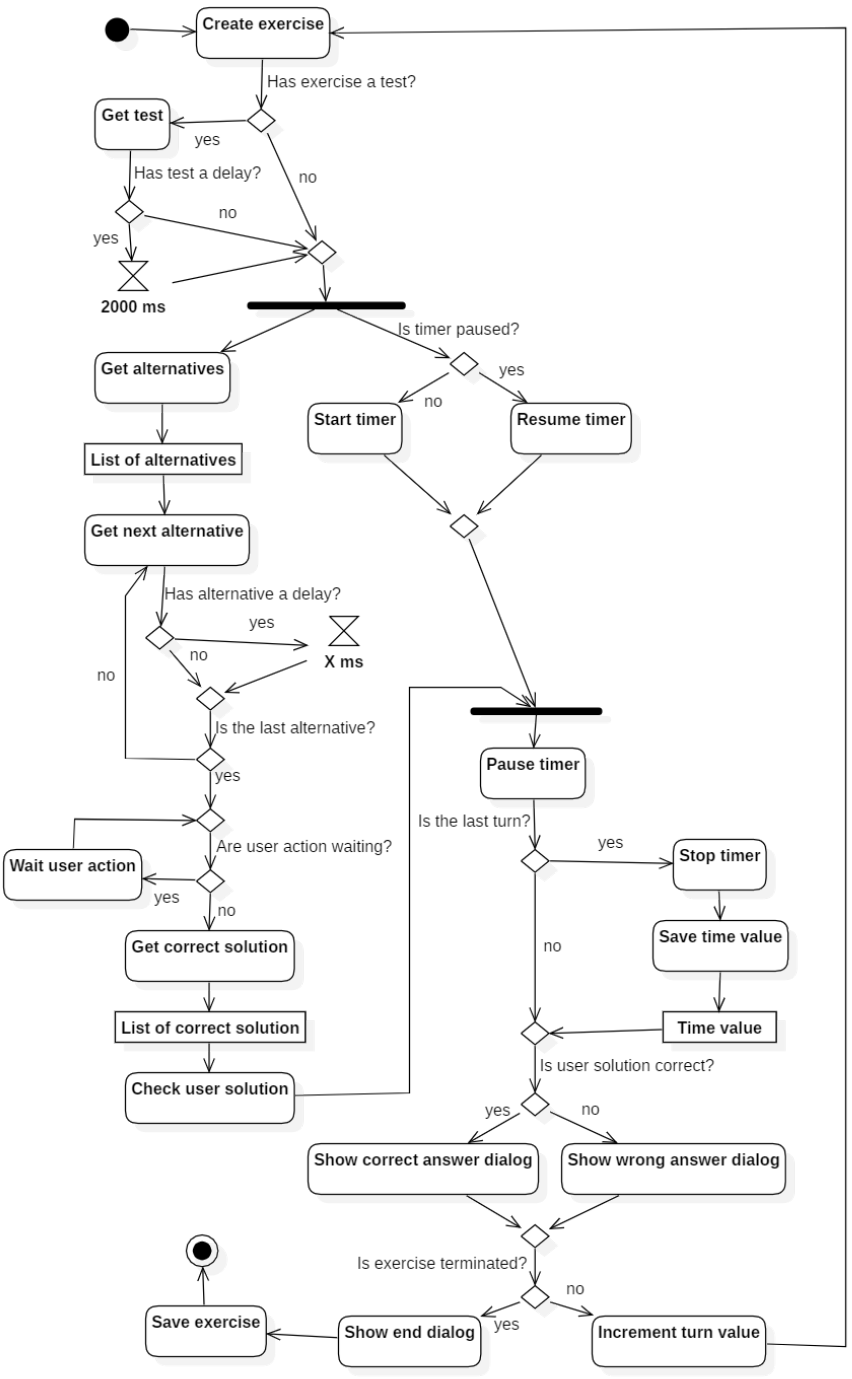


Figura 3.46: Diagramma delle attività che mostra il funzionamento di un esercizio.

3.6.3 Gestione dell'accesso agli esercizi

A seguito della scelta di sviluppare il progetto in linguaggio Java, è stato progettato l'accesso alle classi degli esercizi. La gestione degli accessi sarà svolta tramite l'API Reflection.

L'*API Reflection* [20] è un'infrastruttura che permette di accedere in modo dinamico alle classi degli oggetti Java e di ispezionarli a runtime, per scoprirne la classe di appartenenza e la sua composizione in termini di metodi e interfacce implementate.

L'utilizzo di questa funzione può portare a rischi, perciò se si ritiene che un'operazione possa essere eseguita senza la reflection, allora si deve preferire evitare di usarla. Tra i rischi che si riscontrano principalmente troviamo:

- la riduzione delle prestazioni, quando il codice che si esegue è richiesto molto spesso;
- problemi per le restrizioni di sicurezza, quando un ambiente ha regole specifiche;
- l'esposizione della struttura interna degli oggetti, perché con la Reflection si ha accesso a tutti i metodi e gli attributi di un oggetto e ciò può portare a problemi di sicurezza, a rendere il codice non funzionale o a distruggere la portabilità.

Questo però non significa che l'introduzione dell'uso delle reflection in un progetto non porti anche a molti benefici. Infatti, grazie a questa infrastruttura si ottengono progetti più facili da manutenere, si minimizzano gli errori, c'è un guadagno nella produttività, si ha standardizzazione e si permette l'estensibilità.

L'API Reflection è comunemente utilizzata dai programmi che richiedono la possibilità di esaminare o modificare il comportamento a runtime delle applicazioni in esecuzione sulla Java Virtual Machine. Inoltre, è una funzionalità fondamentale se si desidera costruire applicazioni a componenti.

Grazie al suo utilizzo all'interno del progetto, se si è a conoscenza della generica classe di un esercizio, si potranno conoscere i suoi metodi e i suoi attributi, invocarne i metodi e creare delle istanze della classe, direttamente durante l'esecuzione del software. Tale infrastruttura sarà utilizzata per istanziare e far eseguire gli esercizi all'interno del client.

Capitolo 4

Strumenti e metodi di sviluppo

La scelta degli strumenti e dei metodi di sviluppo da adottare per implementare il progetto è una decisione molto importante, perché ha ripercussioni sulla produttività degli sviluppatori e la qualità del codice.

Qui di seguito verranno illustrati i principali strumenti e metodi di sviluppo adottati, focalizzandosi principalmente sul controllo di versione distribuito adottato, sulle operazioni di build automation e sui principali strumenti utilizzati per lo sviluppo del codice.

4.1 Distributed Version Control System

Un *Distributed Version Control System* (DVCS) è una tipologia di controllo di versione che ha l'obiettivo di condividere le modifiche e le versioni del codice sorgente del software. In questo modo, gli sviluppatori possono collaborare individualmente e parallelamente allo sviluppo di un progetto.

L'utilizzo dei metodi tradizionali non è sbagliato o da evitare, perché aiuta ad eseguire il backup, tracciare e sincronizzare i file, ma con progetti in continua crescita si possono avere dei problemi quando si devono svolgere fusioni o ramificazioni per aggiungere nuove funzionalità al progetto. Un DVCS, invece, facilita la gestione dei cambiamenti, perché non richiede strategie di coordinamento per mantenere la coerenza all'interno di un progetto dopo che sopraggiungono modifiche al codice.

Il principale vantaggio che si ottiene dall'adozione di questo controllo di versione riguarda le prestazioni, perché si permette a centinaia o migliaia di sviluppatori di contribuire allo stesso progetto in contemporanea, avendo a disposizione la cronologia completa delle modifiche (vedi Figura 4.1).

Proprio per via dei vantaggi che si riscontrano con l'utilizzo di un DVCS su un progetto, è stato deciso di fare ricorso ad esso per lo sviluppo di "Train

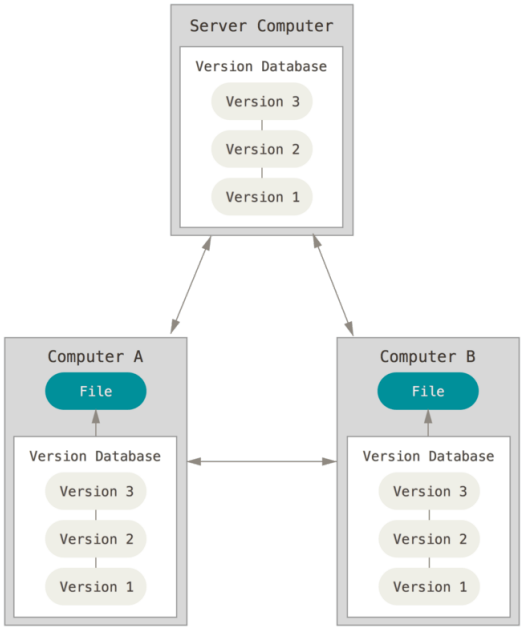


Figura 4.1: Schema del funzionamento di un distributed version control.

your mind", anche se non si aveva a disposizione un team di sviluppatori. In questo modo, conoscendo la posizione del repository, sarà possibile, per chiunque voglia contribuire all'ampliamento del progetto, aggiungere nuove funzionalità.

Per questo progetto, il DVCS scelto è Git [8], uno dei più popolari software di controllo di versione distribuito open source esistenti attualmente.

Git è stato creato nel 2005 da Linus Torvalds per sostituire il DVCS proprietario BitKeeper, che la comunità di sviluppo di Linux stava utilizzando. I suoi vantaggi sono:

- velocità;
- semplicità nell'utilizzo;
- forte supporto per lo sviluppo non lineare (permette molti branch);
- completamente distribuito;
- permette di gestire progetti di grandi dimensioni in modo efficiente.

Il funzionamento di Git è molto semplice, ogni volta che si fa commit o si salva lo stato del progetto, Git fa un'istantanea di come appaiono i file in

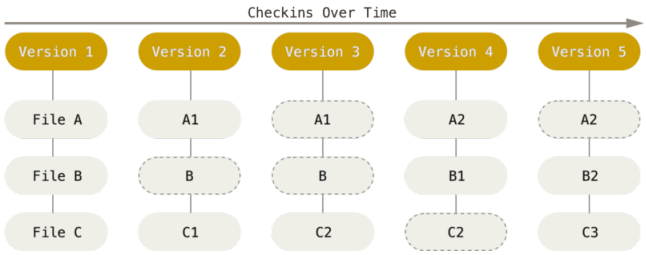


Figura 4.2: Memorizzazione dei dati con istantanee del progetto nel tempo tramite Git.

quel momento, come mostrato in Figura 4.2, senza però memorizzare i file che non hanno subito modifiche. Questa è la peculiarità che differenzia Git da un qualsiasi altro VCS.

Entrando più nello specifico, all'interno del progetto è stato adottato un modello di branching per Git ideato da Vincent Driessen chiamato Gitflow. Questo modello, infatti, è il più adatto alla collaborazione e al ridimensionamento del team di sviluppo.

Per il progetto attuale si prevedono, dunque, due branch principali, con una struttura simile a quella di Figura 4.3, dove:

- il *master* è il branch in cui sarà presente il progetto pronto per la produzione;
- il *develop* è il branch dove saranno presenti le ultime modifiche apportate al progetto per la prossima versione.

4.2 Build automation

Lo sviluppo di un software richiede lo svolgimento di una serie di attività accessorie, come la compilazione del codice, il packaging dei binari, l'esecuzione dei test automatizzati, il deployment della soluzione e la documentazione del progetto.

Gli strumenti di *build automation* sono pensati per automatizzare tutte le fasi di sviluppo del software, perciò hanno il compito di convertire i file e le risorse in un prodotto software nella sua forma finale. Il loro utilizzo è un prerequisito per un uso efficace della continuous integration. Le principali caratteristiche di questi strumenti sono:

- svolgono operazioni per il build dei file, cioè per la trasformazione dei file sorgente nella soluzione finale;

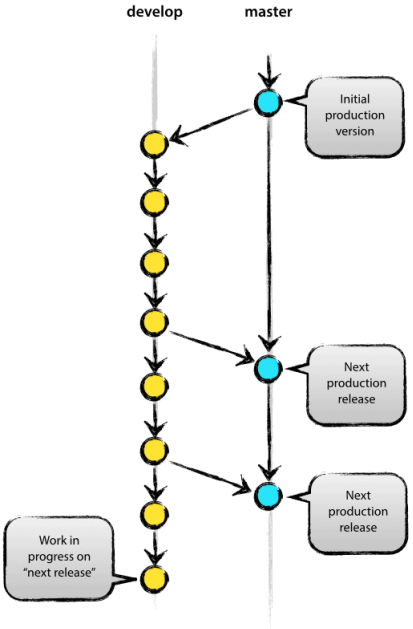


Figura 4.3: Diagramma[1]che illustra la struttura dei branch del progetto.

- organizzano le operazioni in task, cioè in singole operazioni, per poter definire il flusso di svolgimento del processo di build;
- gestiscono le dipendenze del software con altri progetti o librerie attraverso la rete, senza costringere il programmatore a dover mettere a disposizione tutti i sorgenti utilizzati. Ciò significa che, lo strumento di build automation scarica in automatico, durante la compilazione, le dipendenze sempre aggiornate o compatibili con il progetto realizzato, per poter far funzionare il programma.

Per usufruire di tutti i vantaggi che porta l'adozione di un sistema di build automation, è stato deciso di servirsene nel progetto. In particolare, il sistema per l'automazione dello sviluppo scelto è stato Gradle.

Gradle [12] è uno strumento di build automation open source, progettato per essere flessibile in modo da poter costruire quasi ogni tipo di software. La sua unica restrizione è che supporta solamente i repository compatibili con Maven e il filesystem. Il motivo che ha indotto ad utilizzare tale strumento all'interno del progetto risiede nelle sue caratteristiche:

[1]https://nvie.com/posts/a-successful-git-branching-model/

- svolge le attività solo quando necessarie, cioè quando input e output cambiano;
- si esegue sulla JVM, perciò rende facile la sua portabilità anche su piattaforme diverse;
- è possibile estenderlo per crearsi il proprio modello;
- è supportato da moltissimi IDE di sviluppo e interagisce con loro;
- permette subito di identificare i problemi di build.

Grazie a Gradle, è stato possibile integrare, alle librerie base di Java, delle librerie esterne che hanno permesso di sfruttare funzionalità già implementate da altri sviluppatori.

Nel Codice 4.1 è possibile vedere il file di configurazione Gradle del progetto. Si può notare come, ogni libreria ha associata la propria versione, in modo da evitare problemi di incompatibilità tra le librerie e la versione del progetto. Dato che il progetto è diviso in moduli, al file Gradle di base, si aggiungono le dipendenze di ciascun modulo per permettere la compilazione dell'interno progetto. Per i singoli moduli, invece, basterà semplicemente indicare le sole dipendenze che ha il modulo con le librerie esterne o altri moduli (Codice 4.2).

```
group projectGroup
version version

allprojects {
  apply plugin: 'java'

    sourceCompatibility = "${jdk8Version}"

    repositories {
        mavenCentral()
        jcenter()
    }

    dependencies {
        compile "com.typesafe.akka:akka-actor_2.12:
                    ${akkaActorVersion}"
        compile "org.slf4j:slf4j-api:${slf4jVersion}"
        runtime "ch.qos.logback:logback-classic:
                    ${logbackVersion}"

        testCompile "junit:junit:${junitVersion}"
    }
```

```
}

dependencies {
    compile project(':server')
    compile project(':client')
}

defaultTasks('clean', 'build')
```

Codice 4.1: File di configurazione Gradle del progetto.

```
dependencies {
    compile "org.apache.derby:derby:${apacheDerbyVersion}"
    compile "org.apache.derby:derbytools:
                ${apacheDerbyToolsVersion}"
    compile "com.google.code.gson:gson:${gsonVersion}"

    compile project(':utils')
}
```

Codice 4.2: File di configurazione Gradle del modulo server del progetto.

Mentre nel file `build.gradle` si inserisce la configurazione, i numeri di versione delle librerie sono posizionati nel file `gradle.properties`.

4.3 Strumenti utilizzati

Gli strumenti utilizzati per sviluppare il codice, che saranno presentati qui di seguito, sono di due tipologie: l'IDE di sviluppo e le librerie esterne integrate nel progetto.

4.3.1 IDE di sviluppo

L'*Integrated Development Environment* (o semplicemente IDE) è un software che supporta lo sviluppo del codice sorgente di un programma, mettendo a disposizione una serie di strumenti e funzionalità utili allo sviluppo e al debugging.

Per lo sviluppo di questo progetto, è stato scelto come IDE IntelliJ, un ambiente sviluppato da JetBrains per il linguaggio Java, che però supporta un numero considerevole di linguaggi di programmazione. Sulla base del framework, offerto da IntelliJ, sono stati realizzati una serie di ambienti di sviluppo, come ad esempio AndroidStudio.

Questo ambiente di sviluppo integra perfettamente gli strumenti di build automation e fornisce un'interfaccia unificata per i principali sistemi di con-

trollo di versione. Un suo punto di forza è che integra perfettamente all'interno del suo ambiente di lavoro SceneBuilder, lo strumento che consente di progettare rapidamente le interfacce utente per le applicazioni in JavaFX, generando automaticamente il codice FXML.

4.3.2 Librerie esterne

Lo sviluppo del progetto ha richiesto l'integrazione di librerie esterne che hanno permesso di fornire ulteriori funzionalità, rispetto a quelle già fornite dalle librerie di base. In questo modo si è costruita una gerarchia delle dipendenze, che è gestita attraverso Gradle.

Qui di seguito, si riportano le principali librerie utilizzate, ognuna delle quali sarà accompagnata da una breve descrizione. In particolare, una delle librerie importate nel progetto è stata indispensabile per poter creare ed eseguire i test sulle classi e i componenti realizzati.

Akka-actor

Akka-actor è la libreria di base di Akka, che mette a disposizione una serie di operazioni per costruire il proprio sistema di attori all'interno dei progetti. Tale libreria può essere integrata con altre librerie di Akka che permettono di svolgere ulteriori operazioni, come ad esempio la coordinazione tra i sistemi di attori, la persistenza e la gestione degli stream.

Questa libreria è stata impiegata per implementare la concorrenza all'interno del progetto e realizzare il sistema ad attori più adeguato.

Derby

Per realizzare il database embedded Apache Derby, sono state utilizzate le librerie `derby` e `derbytools`.

La libreria `derby` è la libreria di base che permette di costruire dei database embedded. Al suo interno contiene il Derby engine e il driver JDBC di Derby Embedded. Dato che la struttura architetturale è di tipo client-server, tale libreria è stata utilizzata solamente nell'entità server.

Ad affiancare la libreria di base c'è `derbytools`, una libreria che consente di eseguire tutti i tools di Derby.

Javafxsvg

Javafxsvg[2] è una semplice libreria che permette di aggiungere il supporto SVG a JavaFX, consentendo l'utilizzo di immagini in formato SVG, come dei normali tipi di immagini.

Tale libreria è stata utilizzata all'interno del client per gestire le icone delle interfacce grafiche. L'utilizzo delle immagini in formato SVG (*Scalable Vector Graphic*) permette di modificare la scala delle immagini senza perdere informazioni, perché le immagini non sono descritte dal colore dei singoli pixel, ma da formule matematiche, che definiscono le forme e i colori dei singoli oggetti che le compongono. In questo modo si rende l'interfaccia grafica più adattiva.

JFoenix

JFoenix[3] è una libreria Java open source che realizza sui componenti JavaFX lo stile Material Design, un design sviluppato da Google per le interfacce grafiche.

Con l'utilizzo di questa libreria è stato possibile aggiungere alle interfacce grafiche realizzate dei componenti che necessitavano di una minore personalizzazione nello stile grafico. È stata sicuramente molto utile per realizzare i form da compilare per creare pazienti e profili di esercitazione.

TilesFX

TilesFX[4] è una libreria JavaFX, che permette di definire dei tile da inserire nelle dashboard. Attualmente, sono disponibili tile con all'interno widget per creare orologi, calendari, switch o grafici.

La libreria è stata adottata per costruire il grafico radar, il quale mostra al medico il livello di allenamento di ciascun paziente, in base alle funzionalità cognitive allenate durante le esercitazioni.

AnimateFX

AnimateFX[5] è una libreria di animazioni pronte per l'uso in applicazioni JavaFX, che si ispira alla libreria di animazioni CSS `animate.css`.

Tale libreria è stata utilizzata per creare alcune animazioni durante l'esecuzione degli esercizi.

[2]https://github.com/codecentric/javafxsvg
[3]https://github.com/jfoenixadmin/JFoenix
[4]https://github.com/HanSolo/tilesfx
[5]https://github.com/Typhon0/AnimateFX

Gson

Gson[6] è una libreria Java open source che permette di convertire gli oggetti Java nella loro rappresentazione JSON e, viceversa, di convertire una stringa JSON nell'equivalente oggetto Java.

Ci si è serviti di questa libreria per realizzare la lettura di un file JSON contenente l'elenco degli esercizi disponibili da inserire all'interno del database.

TestFX

TestFX[7] è una libreria che permette il test dei componenti grafici di JavaFX. Per poterla utilizzare si deve almeno avere la versione 8 della JDK.

Questa libreria è stata utile per testare il funzionamento di alcuni esercizi del framework, che utilizzano dei componenti grafici di JavaFX. Infatti, essa crea un ambiente applicazione, che permette di simulare il funzionamento di ciascun componente di JavaFX, cosa che invece non accade se semplicemente si utilizza JUnit per eseguire i test.

[6] https://github.com/google/gson
[7] https://github.com/TestFX/TestFX

Capitolo 5

Implementazione

Successivamente alla progettazione e definizione dell'architettura, si è passati all'implementazione vera e propria del progetto. Durante questa fase è stato realizzato il progetto fisico nel rispetto delle scelte prese in precedenza.

Prima di avviare la fase implementativa, è stato scelto il linguaggio di programmazione da utilizzare, il quale doveva permettere di rispettare il requisito di utilizzo di tecnologie note e standard, come già successo per altre decisioni prese in fase di progettazione. La scelta è ricaduta sul linguaggio Java, perché è uno dei linguaggi di programmazione più popolari nel mondo. Nel giugno 2019, questo linguaggio si classifica al primo posto secondo l'indice TIOBE[1] (calcolato in base al numero di ricerche svolte nei motori di ricerca) e al secondo posto secondo l'indice PYPL[2] (calcolato in base al numero di ricerche svolte su Google dei tutorial di ogni linguaggio). Lato client, si è deciso di utilizzare il software applicativo JavaFX, per ottenere un'interfaccia grafica user-friendly, che si ispirasse al mondo web.

L'implementazione ha previsto lo svolgimento di più fasi, che hanno permesso di ottenere come risultato finale il software. Queste fasi sono:

- l'implementazione del modello;
- l'implementazione del database;
- l'implementazione dell'interfaccia grafica;
- l'implementazione del sistema ad attori con la relativa comunicazione;
- l'implementazione del framework degli esercizi e del loro funzionamento nel client.

[1] https://www.tiobe.com/tiobe-index/
[2] http://pypl.github.io/PYPL.html

Qui di seguito saranno messe in evidenza le principali scelte strategiche, che hanno permesso di sviluppare le funzionalità del sistema richieste.

5.1 Gestione del database

A seguito delle decisioni prese in fase di progettazione del database e di definizione della struttura delle tabelle, è stato possibile procedere con la sua implementazione all'interno del progetto.

5.1.1 Gestione della connessione

Per potersi connettere in modo facile al database relazionale scelto, Apache Derby, ci si è avvalsi dell'API JDBC (*Java Database Connectivity*). Il suo utilizzo permette di ottenere l'indipendenza dal DBMS scelto, così che si possa migrare in modo facile a un altro database, senza essere costretti a modificare il codice dell'applicazione. JDBC ha permesso di svolgere tre operazioni:

- stabilire una connessione con il database e accedere alle sue tabelle;
- inviare istruzioni SQL;
- elaborare i risultati ricevuti dal database.

La gestione della connessione al database avviene tramite l'interfaccia `DbConnection` (vedi Figura 5.1), la quale permette di aprire e chiudere la connessione con il database, di creare le tabelle sul database se non esistono e di restituire la classe del DAO richiesta.

Per aprire la connessione al database, per prima cosa, si registra il driver del database, poi si apre la connessione. Il database dovrà avere una posizione precisa all'interno del file system e delle specifiche proprietà. Nel nostro caso, come visibile dal Codice 5.1, l'indirizzo del database sarà una directory locale della macchina su cui si esegue il programma e avrà tre principali proprietà: uno username, una password e dovrà crearsi il database se questo non esiste ancora.

```
1 @Override
2 public Connection openConnection()
3                 throws DbConnectionException {
4   try {
5     if (this.connection == null
6             || this.connection.isClosed()) {
7       Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

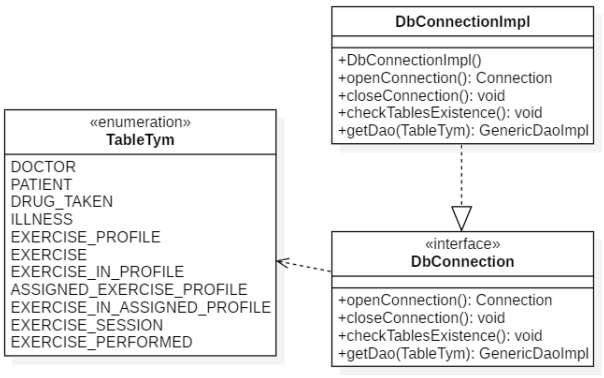


Figura 5.1: Diagramma delle classi per la connessione al database.

```
8         final Properties properties = new Properties();
9         properties.put("create", "true");
10        properties.put("user", USERNAME);
11        properties.put("password", PASSWORD);
12        this.connection = DriverManager
13          .getConnection("jdbc:derby:" + ADDRESS, properties);
14      }
15    } catch(final SQLException e) {
16      throw new DbConnectionException(ERROR_OPEN_CONNECTION);
17    }
18    return this.connection;
19  }
```

Codice 5.1: Codice per l'apertura della connessione con il database.

La creazione delle tabelle, invece, richiede di controllare, tabella per tabella, la loro esistenza all'interno del database. Per poter fare questo controllo è necessario conoscere i metadata del database e verificare se all'interno del suo catalogo esiste una tabella con uno specifico nome. Se nel risultato, restituito con `ResultSet`, non è presente la descrizione della tabella cercata, allora vorrà dire che la tabella non esiste e dovrà essere creata. La creazione avviene tramite l'esecuzione di un comando SQL. Il codice che permette questa operazione è visibile in Codice 5.2.

```
1  @Override
2  public void checkTablesExistence()
3        throws DbConnectionException, DaoException,
4            DbQueryException {
5    for (final TableTym tableTym : TableTym.values()) {
6      openConnection();
```

```
7       try {
8         final DatabaseMetaData metaData =
9                 this.connection.getMetaData();
10        final ResultSet result = metaData
11                .getTables(null, null, tableTym.name(), null);
12        if (!result.next()) {
13          getDao(tableTym).createTable();
14          ...
15        }
16      } catch (final SQLException e) {
17          throw new DbQueryException(ERROR_INSERT_TABLES);
18      }
19      closeConnection();
20    }
21  }
```

Codice 5.2: Codice per la verifica dell'esistenza della tabella e della sua creazione.

Per costruire la classe del DAO richiesto è stato applicato il pattern Factory Method. Tale pattern definisce l'interfaccia per creare gli oggetti, ma poi lascia la decisione del tipo di classe da istanziare alle sottoclassi. In questo caso, come mostrato nel Codice 5.3, si restituisce una tipologia di `GenericDaoImpl` in base alla richiesta fatta dal server, per poter accedere alla tabella desiderata.

```
1  @Override
2  public GenericDaoImpl getDao(final TableTym table)
3                                  throws DaoException {
4    switch (table) {
5      case ASSIGNED_EXERCISE_PROFILE:
6        return new AssignedExerciseProfileDaoImpl(this);
7      case DOCTOR:
8        return new DoctorDaoImpl(this);
9      case DRUG_TAKEN:
10       return new DrugTakenDao(this);
11     ...
12     default:
13       throw new DaoException(ERROR_DAO);
14   }
15 }
```

Codice 5.3: Metodo che applica il pattern Factory Method per restituire la classe del DAO corretta.

5.1.2 Implementazione delle tabelle

Ogni tabella del database è stata rappresentata con una classe, la quale deve implementare la classe astratta `GenericDaoImpl`. In questa classe astratta sono presenti tutti i metodi di base necessari per svolgere le operazioni sulle singole tabelle del database. I metodi previsti sono:

- `createTable`: metodo che esegue la query SQL "CREATE TABLE" per creare la tabella nel database.
- `save`: metodo che esegue la query "INSERT INTO" per inserire un nuovo record sul database.
- `update`: metodo che esegue la query "UPDATE" per aggiornare un record già presente nel database.
- `delete`: metodo che esegue la query "DELETE FROM" per eliminare uno o più record dal database.
- `get`: metodo che esegue la query "SELECT" per ottenere uno o più record nel database con uno specifico identificatore.
- `getAll`: metodo che esegue la query "SELECT" per ottenere tutti i record.

Tutti questi metodi richiedono delle conoscenze, come ad esempio il nome della tabella o i suoi campi, che non sono comuni a tutte le tabelle degli oggetti del sistema. Pertanto, è stato necessario definire dei metodi astratti, che devono essere implementati dalle classi delle singole tabelle. Due metodi astratti rilevanti sono:

- `fillStatement`: metodo che imposta uno statement SQL precompilato con i dati di un oggetto, passati in input.
- `extractModelFromResultSet`: metodo che, preso un `ResultSet` con il risultato di un'operazione sul database, estrae tutti i dati di uno specifico oggetto del modello, per poterlo utilizzare all'interno del programma.

Ogni classe delle tabelle è, anche, corredata da una propria interfaccia, che definisce metodi, come quello di Codice 5.4, che sono applicabili solo a quella specifica tabella.

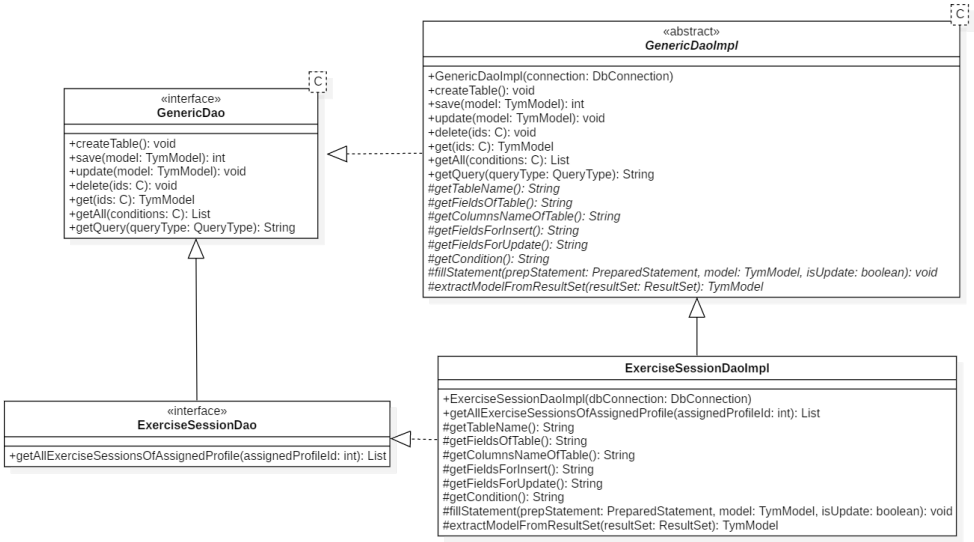


Figura 5.2: Diagramma delle classi per l'implementazione della tabella delle sessioni di esercitazione.

```
@Override
public List<ExerciseSession>
    getAllExerciseSessionsOfAssignedProfile
      (int assignedProfileId) throws DbConnectionException,
                                     DbQueryException {
  return getAll(assignedProfileId)
          .stream()
          .map(es -> (ExerciseSession) es)
          .collect(Collectors.toList());
}
```

Codice 5.4: Metodo che permette di ottenere l'elenco delle sessioni di esercitazione svolte con uno specifico profilo di esercitazione.

Un esempio di implementazione di una tabella del database è presentato in Figura 5.2.

5.1.3 Gestione delle eccezioni

La gestione delle eccezioni permette di definire un insieme di costrutti e regole con lo scopo di gestire in modo più semplice e chiaro il verificarsi di situazioni anomale durante l'esecuzione del programma.

All'interno del progetto è stato deciso di definire alcune eccezioni per permettere al server di gestire, in modo appropriato, il verificarsi di certi errori durante l'esecuzione di alcune operazioni sul database. Le tre eccezioni definite sono:

- `DbConnectionException`: eccezione lanciata quando si verifica un errore durante l'apertura o la chiusura della connessione con il database.
- `DaoException`: eccezione lanciata quando si verifica una situazione anomala durante la creazione della classe che gestisce una tabella per il pattern DAO.
- `DbQueryException`: eccezione lanciata quando si verifica un errore durante l'esecuzione di una query sulle tabelle del database.

Nelle classi `DbConnectionImpl`, `GenericDaoImpl` e in tutte le classi del DAO, le eccezioni sono state gestite per prendere nel client le giuste contromisure al loro verificarsi.

5.2 Interfaccia grafica

Come detto all'inizio del capitolo, l'interfaccia grafica di "Train your mind" è stata realizzata avvalendosi del software applicativo JavaFX.

Ogni schermata consiste in un abbinamento di due oggetti: una view, definita all'interno di un file FXML, e un controller, che gestirà l'aggiornamento e l'interazione dell'utente con la view.

Per poter mostrare le view sullo schermo è necessario caricare il file FXML all'interno del contenitore principale dell'applicazione, chiamato primary stage. Un esempio di caricamento di una view è mostrato nel Codice 5.5, in cui è possibile vedere anche come il file FXML deve essere ricercato tra le risorse, perché non è un file sorgente.

```
1 final FXMLLoader loader = new FXMLLoader();
2 loader.setLocation(getClass()
3         .getResources("/it/unibo/tym/view/BaseLayout.fxml"));
4 final AnchorPane base = loader.load();
5 final Scene scene = new Scene(base);
6 this.primaryStage.setScene(scene);
```

Codice 5.5: Esempio che mostra come si carica una view sul primary stage all'interno del progetto.

La personalizzazione di tutti i componenti grafici è stata realizzata applicando all'intero progetto un foglio di stile, in cui, tramite il CSS, è stato possibile definire i colori, le dimensioni e gli stili di ogni singolo componente.

L'interfaccia grafica ottenuta è possibile vederla in Figura 5.3, dove sono mostrate le schermate principali delle tre sezioni previste.

Qui di seguito verranno descritte le implementazioni di parti rilevanti dell'interfaccia.

5.2.1 Inserimento di un nuovo paziente

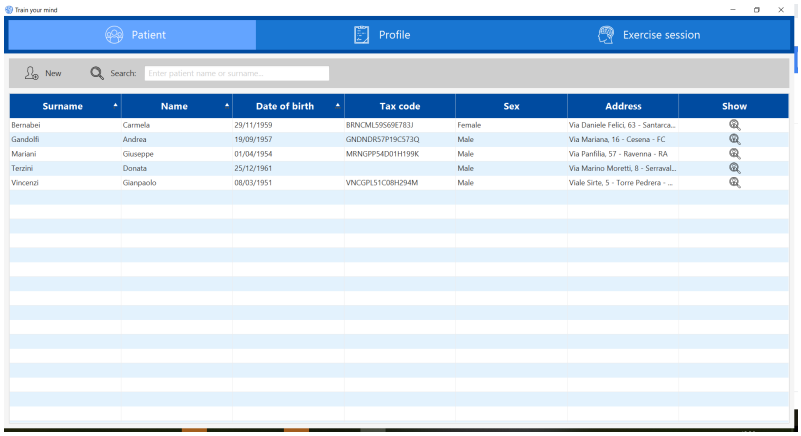
Per inserire un nuovo paziente, si devono compilare una serie di campi con le informazioni personali e cliniche del paziente, che permettano di costruirne la scheda clinica.

La dialog, che si apre durante questa operazione, è quella in Figura 5.4. I dati sono divisi in cinque sezioni:

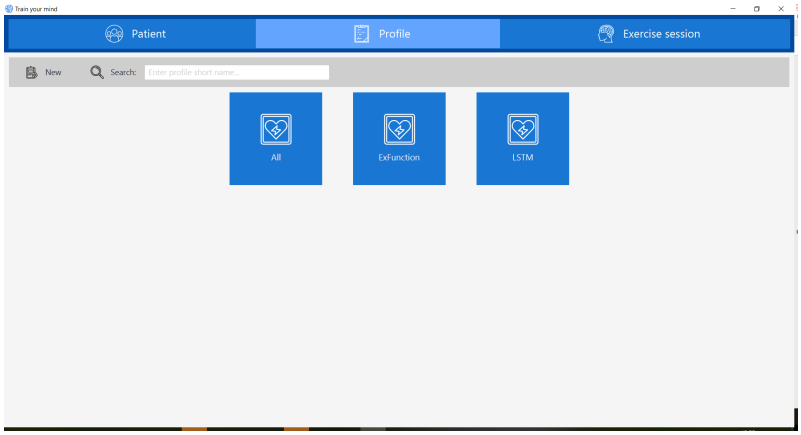
- *Personal data*: sezione in cui si inseriscono i dati personali dei pazienti, come nome, cognome e data di nascita.
- *Address*: sezione in cui si inseriscono i dati riguardanti i recapiti, come l'indirizzo e il numero di telefono.
- *Education and Abilities*: sezione in cui si inseriscono i dati riguardanti il livello di istruzione, il lavoro svolto e il tipo di manualità.
- *Clinical data*: sezione in cui si inseriscono tutti i dati clinici del paziente, come la quantità di alcool, i medicinali assunti e le patologie riscontrate.
- *Note*: sezione in cui il medico può inserire delle note riguardanti il paziente oppure le sedute terapeutiche svolte.

Quando avviene il salvataggio dei dati, si deve controllare la correttezza dei campi compilati. Questo controllo viene svolto anche durante la compilazione del form, grazie all'associazione di un validatore per ogni singolo campo.

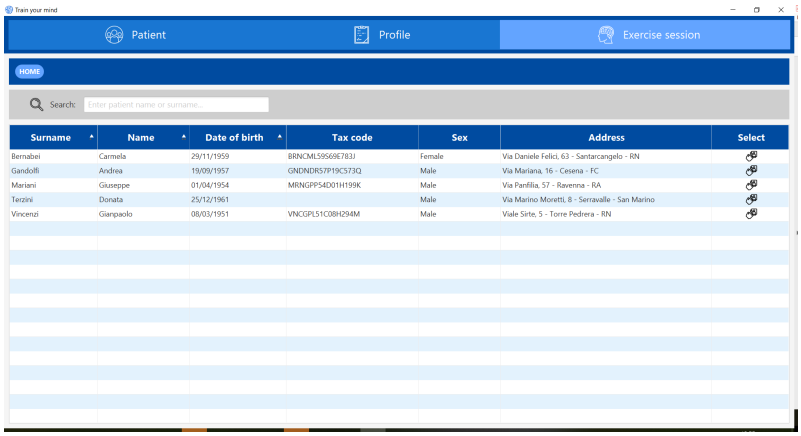
Il validatore è disponibile grazie all'uso della libreria esterna JFoenix, la quale mette a disposizione un validatore di base (`ValidatorBase`), che può essere esteso, e un validatore per controllare se il campo obbligatorio non è stato compilato. Nel progetto sono stati creati quattro nuovi validatori: uno per controllare la correttezza dell'email inserita, uno per la correttezza del codice fiscale, uno per la correttezza del numero telefonico e uno per la correttezza della lunghezza della stringa.



(a) Sezione pazienti.



(b) Sezione profili.



(c) Sezione sessioni di esercitazione.

Figura 5.3: Screenshot delle schermate principali delle tre sezioni della GUI.

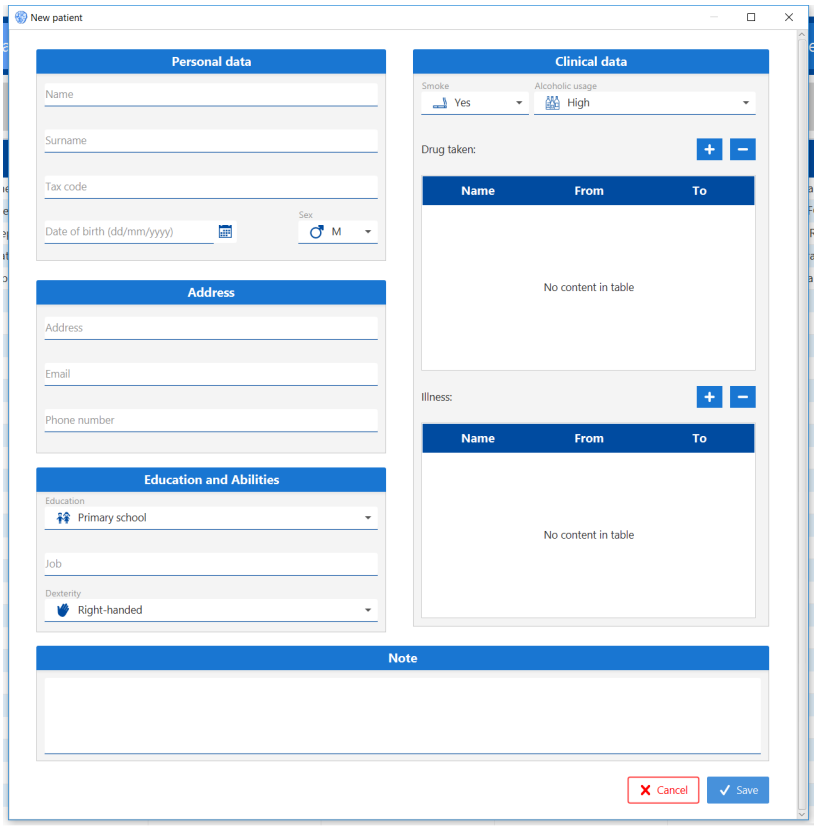


Figura 5.4: Screenshot della dialog per inserire i dati di un nuovo paziente.

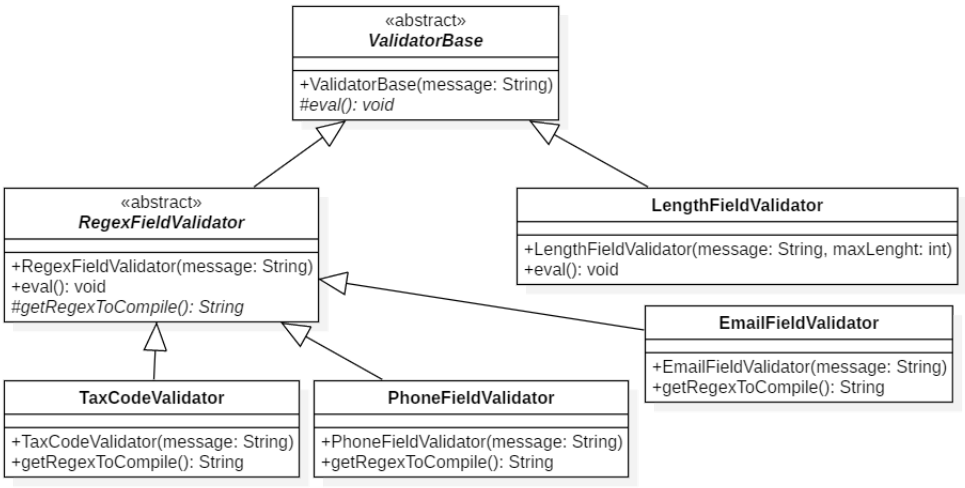
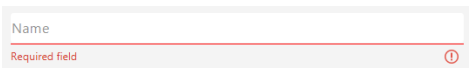
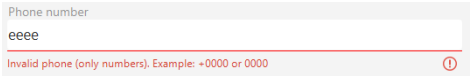


Figura 5.5: Diagramma che mostra la gerarchia dei validatori utilizzati nei campi dei form.



(a) Validatore per un campo obbligatorio.



(b) Validatore personalizzato.

Figura 5.6: Screenshot dei campi di un form con in esecuzione il validatore.

Come mostrato nel diagramma di Figura 5.5, tre validatori utilizzano le espressioni regolari per valutare l'input inserito. Per evitare la duplicazione di codice, è stata quindi creata una classe astratta `RegexFieldValidator`, che implementa il metodo di valutazione, ma che richiede, a chi la estende, di comunicare l'espressione regolare da utilizzare. Un esempio del risultato che si ottiene con l'uso di questi validatori nell'interfaccia è mostrato nella Figura 5.6.

L'aggiunta e la rimozione di medicinali e patologie dal profilo di un paziente è possibile grazie alla presenza di due pulsanti sopra le tabelle di ciascun dato clinico. Dato che la vista per l'inserimento, mostrata in Figura 5.7, è sempre la stessa per entrambe le tipologie di dato, si dovrà indicare al controller in quale delle due tabelle inserire la nuova informazione.

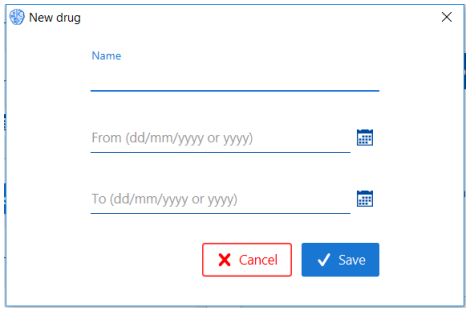


Figura 5.7: Screenshot della dialog per inserire i medicinali assunti o le patologie sofferte dal paziente.

5.2.2 Generazione combo box personalizzate

All'interno di molte interfacce è stato necessario inserire delle combo box per permettere all'utente di effettuare una scelta vincolata agli elementi presenti al loro interno.

Queste combo box dovevano settare gli elementi, estrapolandoli da delle enumerazioni. Così facendo sarebbero però stati mostrati i singoli valori delle enumerazioni e non il testo in base alla localizzazione.

È stato, quindi, necessario definire dei nuovi tipi di celle da impostare nelle combo box. Le nuove tipologie di celle estendono tutte la classe astratta `TymListCell`, che a sua volta estende `ListCell`. Tale classe astratta implementa il metodo `updateItem` (come mostrato nel Codice 5.6), che permette di personalizzare la cella di una lista indicando cosa si vuole mostrare e in quali occasioni.

```
@Override
protected void updateItem(T item, boolean empty) {
  super.updateItem(item, empty);
  String name = null;
  if (item != null && !empty) {
    name = getItemString(item);
    setGraphic(getItemIcon(item));
  } else {
    setGraphic(null);
  }
  this.setText(name);
}
```

Codice 5.6: Metodo che aggiorna un elemento nella lista della combo box.

Ogni classe che estende `TymListCell` deve, poi, implementare i due metodi astratti:

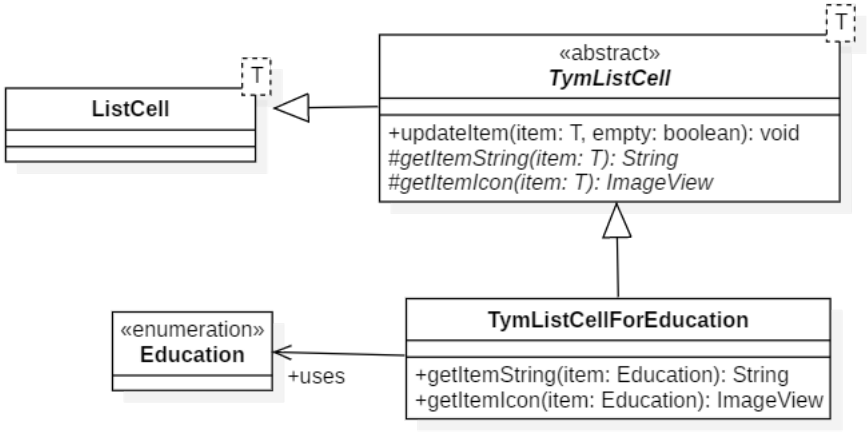


Figura 5.8: Diagramma delle classi per una combo box per il livello di istruzione di un paziente.

- `getItemString`: metodo che ritorna la stringa di testo da mostrare, in base alla localizzazione;
- `getItemIcon`: metodo che ritorna l'icona da settare nella cella, recuperandola dalle risorse.

Il diagramma in Figura 5.8, mostra la classe `TymListCell` e i metodi che mette a disposizione, insieme a un esempio di implementazione.

5.2.3 Creazione di un profilo di esercitazione

Come già fatto per l'inserimento di un nuovo paziente, viene messa a disposizione dell'utente un form (vedi Figura 5.9) da compilare per poter creare un nuovo profilo di esercitazione.

L'elenco degli esercizi disponibili e degli esercizi assegnati al profilo sono gestiti tramite due `ListView`. Le principali operazioni sono però svolte sulla lista degli esercizi assegnati, in cui si può:

- aggiungere un nuovo esercizio, selezionandolo dalla lista di sinistra;
- rimuovere un esercizio assegnato, selezionandolo dalla lista di destra;
- riordinare gli esercizi assegnati.

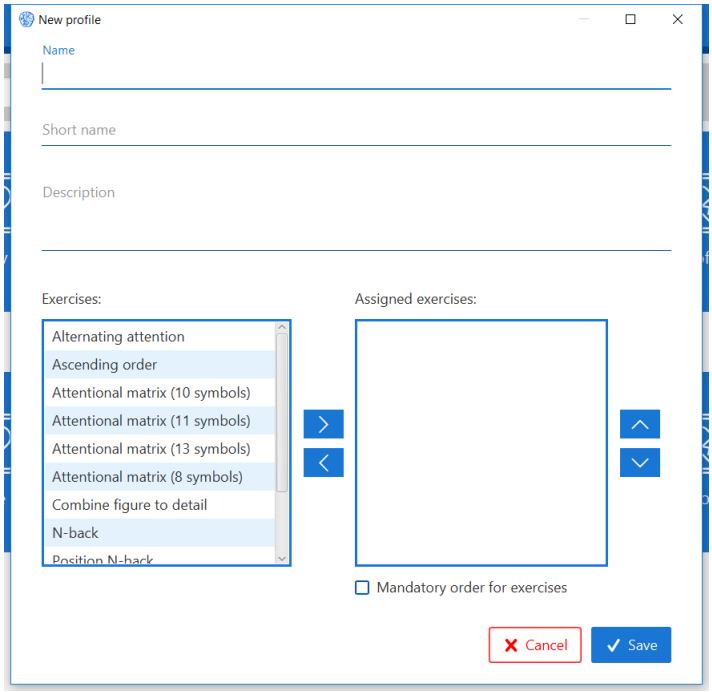


Figura 5.9: Screenshot della dialog per la creazione di un nuovo profilo di esercitazione.

L'implementazione dell'ordinamento degli esercizi all'interno della lista, visibile nel Codice 5.7, prevede di individuare la nuova posizione dell'esercizio nella lista (prima o dopo di quella corrente, a seconda se l'esercizio è da spostare sopra o sotto), rimuovere l'esercizio dalla lista e ricollocarlo nella nuova posizione.

```
private void moveItemInListView(boolean isUpAction) {
  int exerciseSelectedIndex = this.assignedExerciseListView
                .getSelectionModel().getSelectedIndex();
  if ((isUpAction && exerciseSelectedIndex > 0)
      || (!isUpAction && exerciseSelectedIndex != -1
      && exerciseSelectedIndex <
      this.assignedExerciseListView.getItems().size() - 1)) {
    ExerciseInProfileWithExercise exerciseSelected =
        this.assignedExerciseListView
            .getSelectionModel().getSelectedItem();
    this.assignedExerciseListView.getItems()
        .remove(exerciseSelectedIndex);
    int newIndex = isUpAction
                   ? (exerciseSelectedIndex - 1)
                   : (exerciseSelectedIndex + 1);
    this.assignedExerciseListView.getItems()
        .add(newIndex, exerciseSelected);
    ...
    this.assignedExerciseListView.getSelectionModel()
        .select(newIndex);
    this.assignedExerciseListView.scrollTo(newIndex);
  }
}
```

Codice 5.7: Metodo per riordinare gli esercizi all'interno della ListView.

5.2.4 Assegnamento di un profilo di esercitazione

L'assegnamento di un profilo di esercitazione a un paziente viene svolta avvalendosi della dialog mostrata in Figura 5.10, dove l'utente può selezionare i profili da assegnare.

A differenza dell'aggiunta degli esercizi a un profilo, i profili assegnati devono essere rimossi da una `ListView` per essere aggiunti all'altra o viceversa, quando non si vuole più assegnare il profilo. L'esempio di come viene svolta l'operazione di aggiunta di un profilo è mostrato in Codice 5.8.

```
private void setAddButton() {
  ...
  this.addButton.setOnAction(event -> {
    int profileSelectedIndex = this.allProfileListView
```

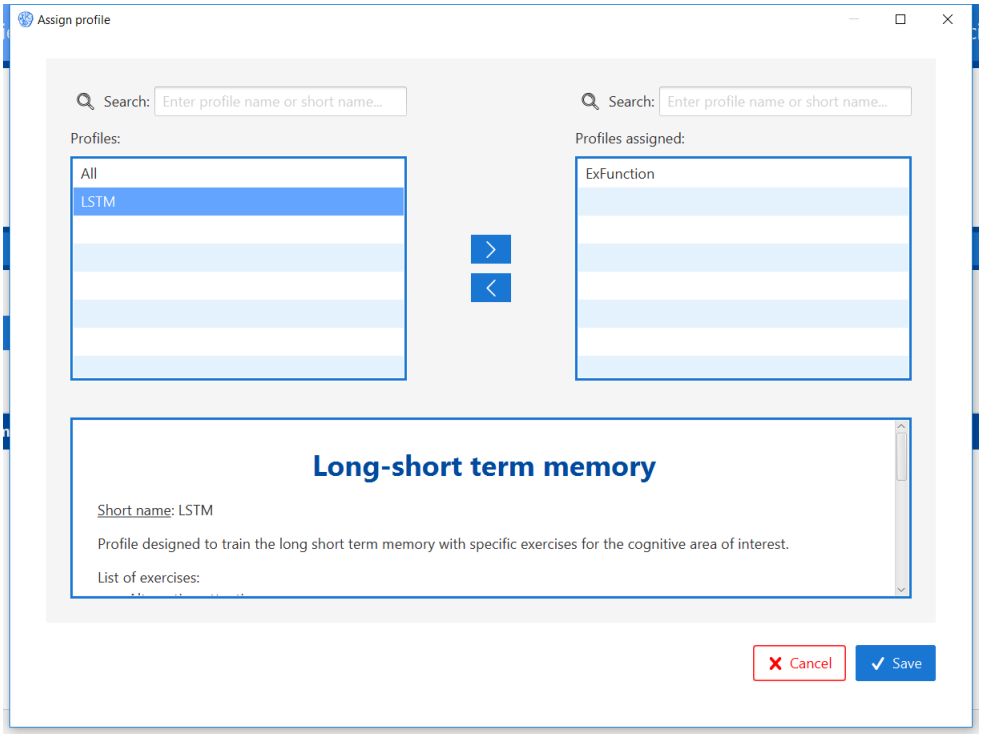


Figura 5.10: Screenshot della dialog per assegnare un profilo di esercitazione a un paziente.

```
        .getSelectionModel().getSelectedIndex();
    if (profileSelectedIndex != -1) {
      ExerciseProfileWithExercise profileSelected =
          this.allProfileListView
          .getSelectionModel().getSelectedItem();
      this.allProfileList.remove(profileSelected);
      this.allProfileListView.getItems()
          .remove(profileSelectedIndex);
      this.profileSelectedList.add(profileSelected);
      sortSelectedProfileList();
    } else {
      ...
    }
  });
}
```

Codice 5.8: Metodo per aggiungere un profilo all'interno della ListView.

Nell'interfaccia, si mostrano anche tutte le informazioni dei profili selezionati ancora da assegnare, per poter aiutare l'utente nella scelta. Tra tutte le informazioni disponibili, si elencano anche gli esercizi assegnati al profilo.

In casi come questo, dove nella vista è disponibile un campo di ricerca che aiuta l'utente a trovare l'oggetto richiesto, si deve assegnare un listener al campo per controllare se nella lista è presente ciò che si sta cercando e per mostrare solo i risultati che corrispondono alla ricerca.

Come mostrato in Codice 5.9, per permettere un confronto con testi di qualsiasi formato, si trasformano tutte le stringe in caratteri minuscoli (metodo `toLowerCase`). Tale stringa trasformata viene poi confrontata con i nomi di interesse (nel caso dei profili con il nome del titolo e il suo nome corto). Il risultato di ricerca viene posizionato in una lista filtrata.

```
private void addListenerToSearchField(TextField field,
      List<ExerciseProfileWithExercise> list,
      ListView<ExerciseProfileWithExercise> listView) {
  ...
  field.textProperty()
      .addListener((observable, oldValue, newValue) -> {
    if (list.size() != 0) {
      listView.getItems().clear();
      ObservableList<ExerciseProfileWithExercise>
        filteredList = FXCollections.observableArrayList();
      if (newValue == null || newValue.isEmpty()) {
        listView.getItems().addAll(list);
      } else {
        String lowerCaseFilter = newValue.toLowerCase();
        for (ExerciseProfileWithExercise profileWithExercise
            : list) {
```

```
17            if (profileWithExercise.getExerciseProfile()
18                  .getName().toLowerCase()
19                    .contains(lowerCaseFilter)
20                || profileWithExercise.getExerciseProfile()
21                  .getShortName().toLowerCase()
22                    .contains(lowerCaseFilter)) {
23              filteredList.add(profileWithExercise);
24            }
25          }
26          listView.getItems().addAll(filteredList);
27        }
28      }
29    });
30  }
```

Codice 5.9: Metodo per ricercare i profili di esercitazione da assegnare.

5.2.5 Generazione di table view personalizzate

Nell'interfaccia grafica è stato fatto largo uso delle `TableView` per mostrare all'utente liste di elementi con le relative informazioni. Le informazioni inserite, però, non hanno sempre un formato idoneo per la stampa a video, perciò è stato necessario definire delle celle che svolgessero un corretto rendering dei dati nella cella della colonna.

Per assegnare un nuovo tipo di cella a una specifica colonna della tabella, è stato utilizzato il metodo `setCellFactory`, a cui si deve passare in input una classe che estende `TableCell`.

Le celle create per il progetto sono mostrate nel diagramma di Figura 5.11 e si possono distinguere in tre tipologie:

- le classi che estendono la classe astratta `FormattedTableCell`, permettono di impostare nella colonna una formattazione particolare del testo. Ad esempio `DateFormattedCell` imposta la stampa della data con il formato "dd/mm/yyyy" (Figura 5.12c) oppure `DurationFormattedCell` imposta la stampa del tempo indicando i minuti e i secondi.

- le classi che estendono `ButtonTableCell`, permettono di impostare nella colonna un pulsante che l'utente può cliccare per svolgere una qualche tipo di operazione. Nel nostro caso, il pulsante aprirà sempre un'altra vista, ma con scopi diversi (alcune volte per mostrare qualcosa come `ButtonTableCellShow` - Figura 5.12a -, altre per selezionare come `ButtonTableCellSelect`).

- `LevelTableCell` permette di impostare nella colonna della tabella delle combo box per selezionare il livello di difficoltà (Figura 5.12b).

5.2.6 Creazione dei grafici

Il medico può avere la necessità di prendere visione l'andamento dell'attività di allenamento del paziente, non solo tramite numeri, ma anche attraverso strumenti visivi, che facciano saltare subito all'occhio se sono necessari aggiustamenti nel training svolto.

Per mostrare l'andamento sono, quindi, stati inseriti dei grafici che permettono di visionare l'andamento del singolo paziente. I grafici realizzati sono di due tipologie:

- grafico a linee (Figura 5.13a);
- grafico radar (Figura 5.13b).

Grafico a linee

Il grafico a linee è stato inserito per permettere al dottore di vedere l'andamento dell'allenamento del paziente nel tempo e su uno specifico compito. L'analisi dell'andamento può essere fatto in quattro modi diversi:

- l'andamento è misurato in base al tempo impiegato per terminare l'esercizio;
- l'andamento è misurato in base al numero di errori commessi durante l'esecuzione dell'esercizio (il numero di errori si ottiene con la somma del numero di omissioni e il numero di falsi allarmi);
- l'andamento è misurato in base al tempo impiegato e al numero di errori commessi. È stato quindi necessario normalizzare per ottenere valori che fossero compresi tra 0 e 1;
- l'andamento è misurato in base al livello di difficoltà assegnato ad ogni esercizio.

Questa tipologia di grafico è stata realizzata istanziando la classe `LineChart`, disponibile nella libreria di Java, nel package `javafx.scene.chart`. Nel Codice 5.10, è possibile vedere come il grafico ha gli assi di due tipologie diverse: nell'asse delle ascisse contiene dati di tipo categorico (sono delle stringhe contenenti la data di svolgimento dell'esercizio) e nell'asse delle ordinate contiene dati di tipo numerico (sono i valori della misura, come il tempo o il numero di errori). Solamente nel caso di grafico con i livelli, anche l'asse delle ordinate è di tipo categorico, perché deve mostrare i valori dei possibili livelli di difficoltà per quell'esercizio.

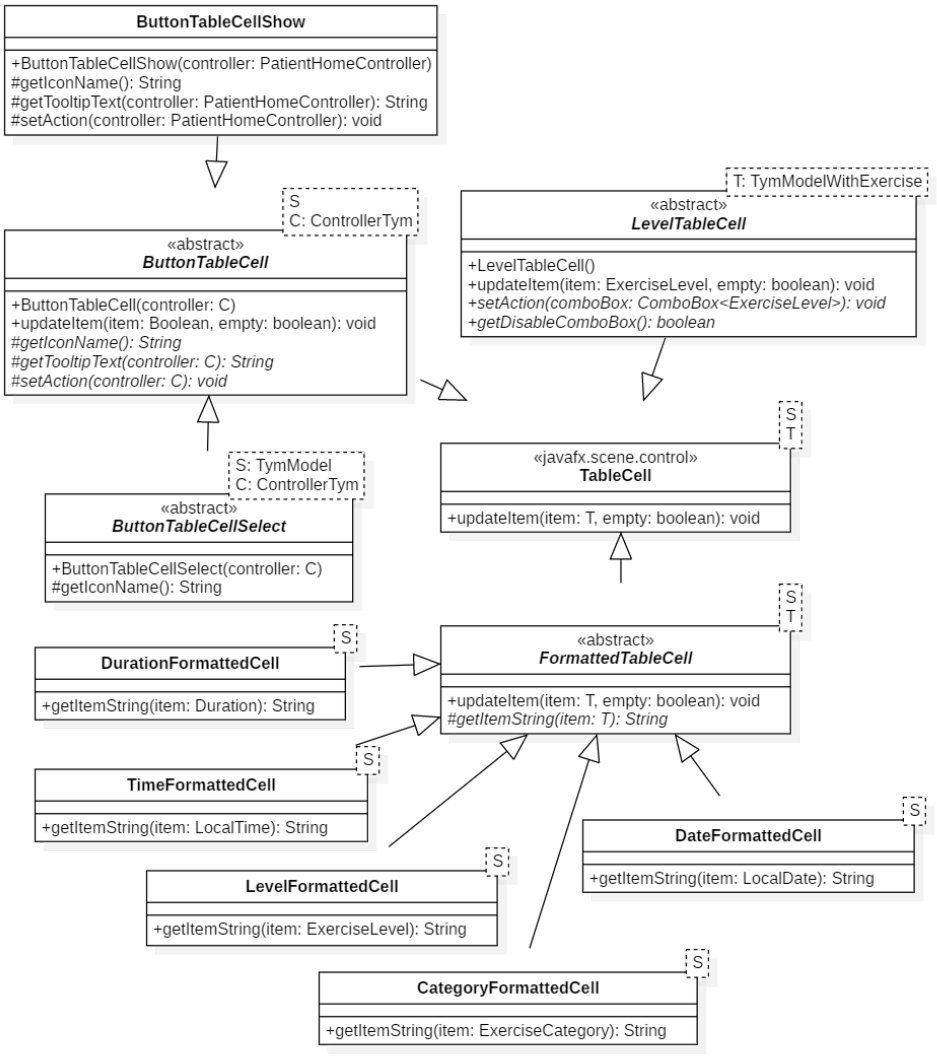
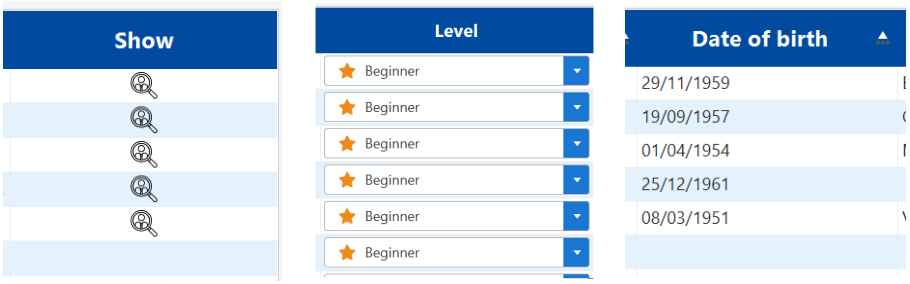


Figura 5.11: Diagramma delle classi con la gerarchia delle `TableCell` utilizzate nel progetto.

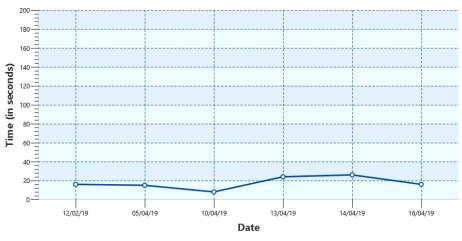


(a) Colonna con `ButtonTableCellShow`.

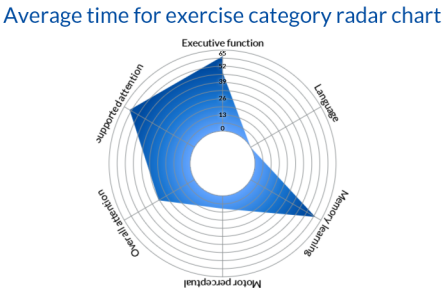
(b) Colonna con `LevelTableCell`.

(c) Colonna con `DateFormattedCell`.

Figura 5.12: Screenshot di alcune colonne delle `TableView` del progetto a cui sono state assegnate alcune delle tipologie di celle create.



(a) Grafico a linee.



(b) Grafico radar.

Figura 5.13: Screenshot dei grafici per mostrare l'andamento dell'allenamento del singolo paziente.

```
private static LineChart<String, Number> setLineChart(
                    String yLabel,
                    double upperBound,
                    double tickUnit) {
  NumberAxis yAxis = new NumberAxis();
  yAxis.setLabel(yLabel);
  yAxis.setAutoRanging(false);
  yAxis.setLowerBound(Y_AXIS_LOWER_BOUND);
  yAxis.setUpperBound(upperBound);
  yAxis.setTickUnit(tickUnit);
  return new LineChart<>(getXAxis(), yAxis);
}

private static CategoryAxis getXAxis() {
  CategoryAxis xAxis = new CategoryAxis();
  xAxis.setLabel(BundleUtils.getBundle("date"));
  return xAxis;
}
```

Codice 5.10: Metodi per impostare il `LineChart` per il tempo impiegato e per il numero di errori commessi.

Una volta ottenuto il grafico con le caratteristiche desiderate, si deve ottenere una `ObservableList<XYChart.Data>`, per poter aggiungere i dati al grafico e mostrare il grafico a linee. Per estrarre i dati utili, è stato realizzato il metodo di Codice 5.11, in cui per ogni esercizio svolto si estraggono la data di svolgimento e un valore, che può essere il tempo impiegato (in secondi) oppure il numero di errori commessi.

```
private static ObservableList<XYChart.Data<String, Number>>
  getDataForChart(
    ExerciseWithExercisesPerformed exerciseSelected,
    long[] minMaxValue,
    boolean isTimeChart,
    List<ExerciseSessionWithExerciseProfile> sessionList) {
  ObservableList<XYChart.Data<String, Number>> data =
                FXCollections.observableArrayList();
  for (ExercisePerformed exercisePerformed
      : exerciseSelected.getExercisePerformedList()) {
    ExerciseSessionWithExerciseProfile exerciseSession =
        sessionList.stream()
              .filter(session
                -> session.getExerciseSession().getId()
                == exercisePerformed.getExerciseSessionId())
              .findAny()
              .orElse(null);
    if (exerciseSession != null) {
      Timestamp startTimestamp = exerciseSession
```

```
20            .getExerciseSession().getStartTimestamp();
21        String date = startTimestamp
22            .toLocalDateTime().toLocalDate()
23            .format(DateTimeFormatter.ofPattern(DATE_FORMAT));
24        double value = 0;
25        if (isTimeChart) {
26          value = exercisePerformed
27                  .getTotalTime().getSeconds();
28        } else {
29          value = exercisePerformed.getNumberOfOmissions()
30                + exercisePerformed.getNumberOfFakeAlarms();
31        }
32        if (minMaxValue != null) {
33           value = (value - minMaxValue[0])
34                / (minMaxValue[1] - minMaxValue[0]);
35        }
36        XYChart.Data<String, Number> newData =
37              new XYChart.Data<>(date, value);
38        ...
39        data.add(newData);
40      }
41    }
42    return data;
43  }
```

Codice 5.11: Metodo per estrarre i dati da inserire nel grafico.

Grafico radar

Il grafico radar, invece, permette all'utente di vedere il tempo medio impiegato per risolvere gli esercizi di ciascuna area cognitiva allenata dal paziente.

Dato che la libreria di Java non mette a disposizione un grafico di questo tipo, è stato necessario trovare una libreria esterna che ne permettesse l'implementazione. Ciò, invece, è stato possibile con la libreria `tilesfx`, che ha permesso di istanziare la classe `Tile`, a cui è stato aggiunto il grafico radar. Il Codice 5.12 permette di vedere come sono stati ricavati i dati da inserire nel grafico e come si è istanziato il grafico radar.

```
List<ChartData> chartDataList = new ArrayList<>();
double maxValue = 0;
...
for (final ExerciseCategory category
              : ExerciseCategory.values()) {
  if (category != ExerciseCategory.TEST
        && category != ExerciseCategory.ALL) {
    double valueOfChart = 0;
    if (exercisePerCategoryValue.containsKey(category)
```

```
10            && exercisePerCategoryOccurrences
11                    .containsKey(category)) {
12        valueOfChart = exercisePerCategoryValue.get(category) /
13              exercisePerCategoryOccurrences.get(category);
14      }
15      if (valueOfChart > maxValue) {
16        maxValue = valueOfChart;
17      }
18      ...
19      ChartData chartData = new ChartData(BundleUtils
20              .getBundle(category.getCategoryBundleName()),
21                    valueOfChart, Tile.BLUE);
22      chartData.setTextColor(Color.BLACK);
23      chartDataList.add(chartData);
24    }
25  }
26  double upperBound = maxValue + OFFSET_UPPER_BOUND_CHART;
27  Tile radarChart = TileBuilder.create()
28      .skinType(Tile.SkinType.RADAR_CHART)
29      .minValue(LOWER_BOUND_CHART)
30      .maxValue(upperBound)
31      .radarChartMode(RadarChart.Mode.POLYGON)
32      .gradientStops(new Stop(0, Color.TRANSPARENT),
33          new Stop(0.00001, Color.web("#63a4ff")),
34          new Stop(0.4, Color.web("#1976d2")),
35          new Stop(0.8, Color.web("#004ba0")))
36      .chartData(chartDataList)
37      .animated(true)
38      .backgroundColor(Color.WHITE)
39      .foregroundColor(Color.BLACK)
40      .title(BundleUtils.getBundle("radar_chart_title"))
41      .titleColor(Color.web("#004ba0"))
42      .titleAlignment(TextAlignment.CENTER)
43      .textSize(Tile.TextSize.NORMAL)
44      .tooltipText(tooltipText)
45      .build();
46  ...
```

Codice 5.12: Stralcio di codice per ottenere i dati da inserire nel grafico radar e per istanziare il grafico.

5.3 Attori e comunicazione

Il sistema ad attori realizzato all'interno del progetto è costituito da due entità: ClientActor e ServerActor.

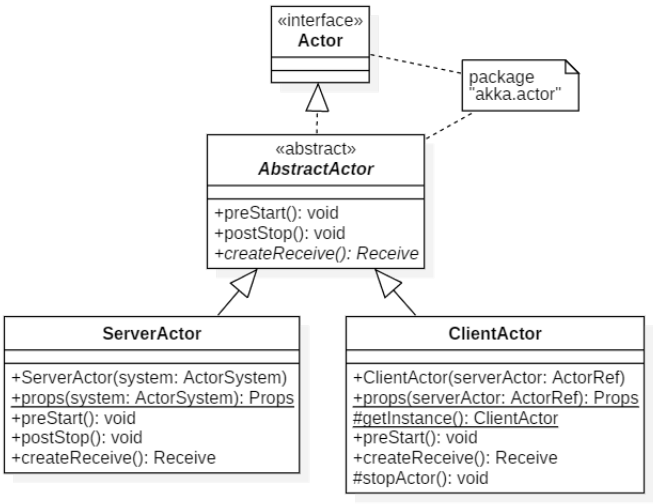


Figura 5.14: Diagramma delle classi con la gerarchia degli attori.

Come visibile nella Figura 5.14, entrambi questi attori sono un'estensione dell'`AbstractActor`, classe presente nella libreria di Akka, che mette a disposizione i metodi per mettere in comunicazione gli attori di un sistema.

Dalle classi di questi due attori, i metodi rilevanti sono:

- `props`: metodo statico che crea l'oggetto di configurazione utile per creare l'attore. Al suo interno specifica le opzioni di creazione dell'attore.
- `preStart`: metodo che contiene tutte le operazioni da svolgere all'avvio dell'attore. All'interno di questo metodo, ServerActor crea il database e le sue tabelle e aggiorna gli esercizi disponibili, mentre ClientActor avvia l'interfaccia grafica.
- `createReceive`: metodo che definisce quali messaggi l'attore può gestire e in che modo questi messaggi devono essere elaborati. Viene realizzato questo comportamento grazie a un `receiveBuilder`.
- `stopActor`: metodo del ClientActor che viene chiamato quando l'interfaccia grafica viene chiusa e serve a comunicare al ServerActor che il suo lavoro deve terminare.
- `getInstance`: metodo del ClientActor che ritorna il riferimento alla classe stessa. Permette ai controller del client di accedere ai metodi dell'attore per inviare le richieste al server.

Oltre ai metodi mostrati, il ClientActor mette a disposizione altri metodi pubblici per permettere ai controller dell'interfaccia grafica di inviare le richieste al server.

Questo significa che, per comunicare, ClientActor e ServerActor si devono scambiare dei messaggi. Ogni messaggio scambiato è rappresentato dall'istanziazione di una classe che rappresenta il messaggio stesso. Le tipologie di messaggi presenti all'interno del progetto sono tre:

- un normale messaggio, che richiede lo svolgimento di un'operazione sul database o comunica il risultato ottenuto dopo lo svolgimento di un'operazione sul database. Un esempio di questa tipologia può essere `PatientMessage`, un messaggio che se ricevuto dal server richiede di inserire le informazioni di un nuovo paziente sul database, mentre se ricevuto dal client comunica l'identificativo del paziente appena inserito sul database.

- un messaggio di conferma, `ConfirmMessage`, che viene sempre inviato dal server al client, per comunicare in certe occasioni che l'operazione richiesta è stata eseguita con successo.

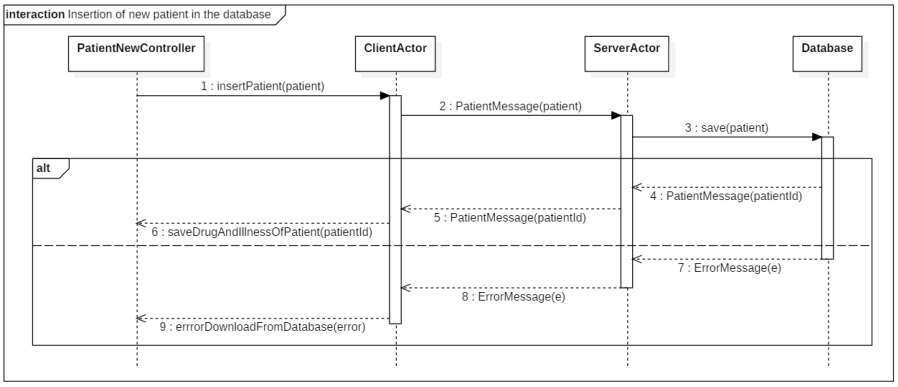
- un messaggio di errore, `ErrorMessage`, che viene sempre inviato dal server al client per comunicare che si è verificato un evento anomalo, perciò il client dovrà agire di conseguenza, mostrando il messaggio di errore all'interno dell'interfaccia.

In ogni caso, il funzionamento della comunicazione tra gli attori del sistema è di tipo RPC-like, come visibile nella Figura 5.15, perché ad ogni richiesta del client, il client rimane in attesa di una risposta da parte del server per poter procedere con le operazioni successive.

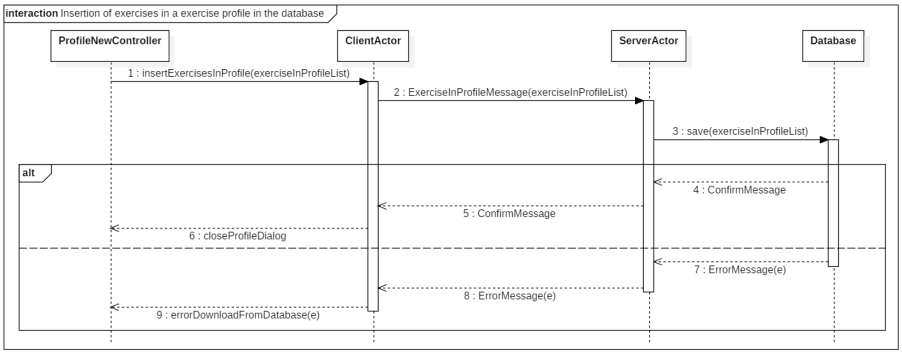
L'actor system è creato dalla classe principale `Launcher`, che ha il compito di avviare l'interno sistema. Come si può vedere nel Codice 5.13, all'avvio del sistema si crea il sistema di attori con un nome, poi, si crea l'attore ServerActor e l'attore ClientActor.

```
1 public static void main(final String[] args) {
2   ActorSystem system = ActorSystem.create(SYSTEM_NAME);
3   ActorRef server = system
4         .actorOf(ServerActor.props(system), SERVER_NAME);
5   system.actorOf(ClientActor.props(server), CLIENT_NAME);
6 }
```

Codice 5.13: Codice di avvio del programma che istanzia il sistema di attori del progetto.



(a) Messaggi scambiati per l'inserimento di un nuovo paziente nel database.



(b) Messaggi scambiati per l'inserimento degli esercizi in un profilo di esercitazione.

Figura 5.15: Esempi di messaggi scambiati tra il ClientActor e il ServerActor.

5.4 Esecuzione degli esercizi

Come già detto nella sottosezione 3.6.3, l'esecuzione di un esercizio, durante una sessione di esercitazione, è un'operazione che è svolta grazie all'utilizzo delle reflection.

Grazie alla loro adozione è possibile istanziare un esercizio solo conoscendo il nome della classe, senza dover dichiarare una variabile per ogni singolo esercizio presente all'interno del framework.

La generazione del compito da eseguire viene svolta tramite le operazioni mostrate nel Codice 5.14, in cui è possibile notare tre azioni principali:

- l'ottenimento della classe dell'esercizio da istanziare tramite il nome della classe;
- l'ottenimento del costruttore della classe dell'esercizio;
- la creazione della nuova istanza dell'esercizio, passando al suo costruttore i parametri desiderati.

Questi passaggi sono necessari, perché alcuni esercizi hanno bisogno di conoscere alcune informazioni utili per poter, poi, generare le alternative e le soluzioni corrette. Ad esempio, i parametri che possono essere passati ai vari costruttori sono la dimensione delle figure geometriche da generare o il seme da passare al generatore di numeri random per avere sempre la stessa combinazione di risultati o il tempo massimo assegnato per svolgere l'esercizio. I costruttori di tutti gli esercizi prendono in input come parametro il livello di difficoltà dell'esercizio.

```
...
this.exerciseClass = Class
    .forName(this.exerciseClassName.getClassName());
Constructor exerciseConstructor;
...
switch (this.exerciseClassName) {
  case ASCENDING_ORDER:
  case COMBINE_FIGURE_TO_DETAIL:
    exerciseConstructor = this.exerciseClass
        .getDeclaredConstructor(ExerciseLevel.class);
    this.exerciseSelected =
        (ExerciseGeneric) exerciseConstructor
            .newInstance(this.exerciseLevel);
    ...
    break;
  ...
  case LETTER_N_BACK:
```

```
18    case SART:
19      exerciseConstructor = this.exerciseClass
20          .getDeclaredConstructor(ExerciseLevel.class,
21              double.class);
22      this.exerciseSelected =
23          (ExerciseGeneric) exerciseConstructor
24            .newInstance(this.exerciseLevel, this.maxTime);
25      ...
26      break;
27    ...
28  }
```

Codice 5.14: Frammento di codice per la generazione dell'esercizio tramite le reflection.

Qui di seguito verranno descritti la vista dell'esercizio da svolgere, le animazioni realizzate, gli eventi catturati e saranno presentate le implementazioni degli esercizi del framework.

5.4.1 Vista

Una volta generato l'esercizio da eseguire si devono settare i componenti grafici all'interno dell'interfaccia. La vista dell'esercizio si compone di tre sezioni:

- una barra delle informazioni;
- una sezione per il test;
- una sezione per le alternative.

Tali sezioni, che nella Figura 5.16 sono ben distinguibili, non sono sempre visibili. La barra delle informazioni è sempre presente, ma la sezione di test e delle alternative possono essere mostrate o meno a seconda del tipo di compito da svolgere.

Barra delle informazioni

La barra delle informazioni (Figura 5.17) è stata progettata per fornire informazioni utili all'utente, durante lo svolgimento di un esercizio. Oltre a presentare il nome dell'esercizio, mette a disposizione altri componenti grafici, che permettono di svolgere alcune operazioni:

- tornare alla schermata delle istruzioni per poter vedere nuovamente come deve essere svolto il compito (pulsante con il punto interrogativo);

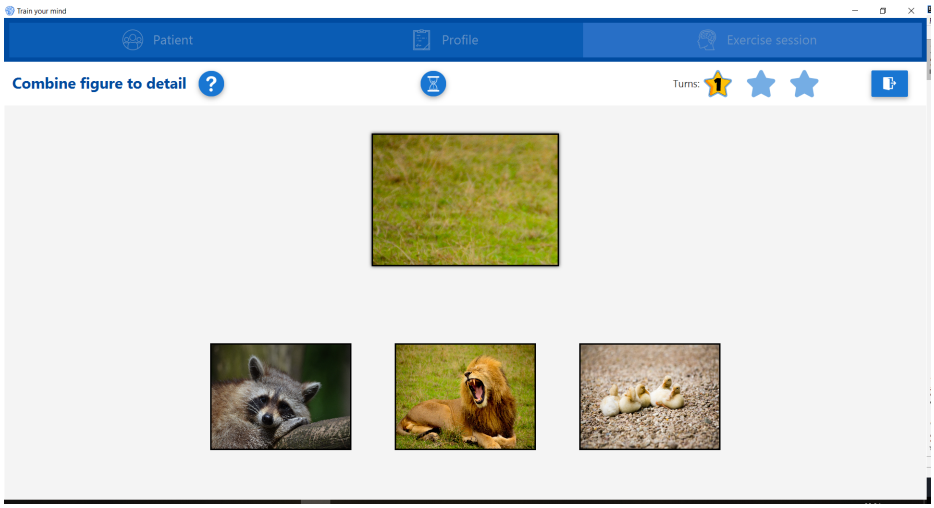


Figura 5.16: Esempio di vista di un esercizio.

Figura 5.17: Barra delle informazioni.

- mostrare la barra del tempo, per vedere quanto tempo è passato e quantificare quanto tempo manca al termine del compito (pulsante con la clessidra);
- vedere il numero di turni di cui si compone l'esercizio e a che punto si trova l'utente (stelle);
- tornare alla schermata con l'elenco degli esercizi da svolgere, annullando tutti i progressi fatti durante l'esecuzione dell'esercizio (pulsante con la porta).

5.4.2 Animazioni

L'esecuzione degli esercizi è stata affiancata dall'utilizzo di diverse animazioni, le quali hanno scandito il corso dell'esecuzione e hanno migliorato la user experience del compito stesso.

Qui di seguito verranno descritte le principali animazioni utilizzate all'interno del progetto.

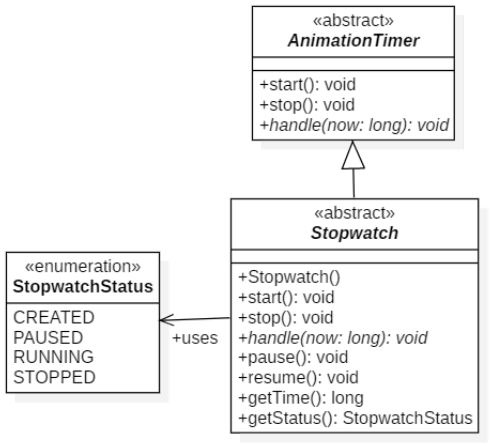


Figura 5.18: Diagramma delle classi per il cronometro.

Stopwatch

Lo stopwatch è il cronometro che scandisce il tempo dell'esercizio e permette di ottenere il tempo impiegato dall'utente per svolgere un determinato compito.

La libreria di JavaFX mette a disposizione la classe `AnimatedTimer` per realizzare un timer, ma non permette di svolgere operazioni come la pausa o il riavvio del timer. Per questo motivo è stata creata la classe `Stopwatch`, la quale estende la classe di JavaFX, ma implementa nuovi metodi utili allo svolgimento dell'esercizio.

Come visibile in Figura 5.18, al cronometro è associato uno stato. Tale stato è necessario per decidere quale operazione compiere sul cronometro in base alle necessità. Ad esempio, se si vogliono vedere nuovamente le istruzioni del compito, il cronometro dovrà essere messo in pausa, ma se il cronometro non è mai stato avviato (non è in stato `RUNNING`), non potrà mai essere messo in pausa.

Il metodo `handle` dovrà essere implementato dalla classe che utilizza il cronometro (`ExerciseController`), in quanto sarà chiamato ad ogni frame, mentre il cronometro è attivo, e in esso dovranno essere indicate le operazioni da svolgere ogni volta che si aggiorna il valore del cronometro.

Nel costruttore di `ExerciseController`, il cronometro è stato definito come mostrato nel Codice 5.15. Ad ogni frame si aggiorna il valore all'interno di una progress bar, poi si controlla se il tempo è terminato, perché a quel punto si deve bloccare l'avanzamento dell'utente nell'esercizio e controllare

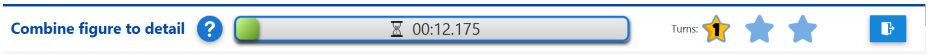


Figura 5.19: Barra degli strumenti con la barra del tempo visibile.

le risposte date.

```
this.timer = new Stopwatch() {
  @Override
  public void handle(long now) {
    long currentTime = getTime();
    ExerciseController.this.timerBar
          .setProgressTimeBarValue(currentTime);
    if ((ExerciseController.this.numTurnTot != 1
        || ExerciseController.this.exerciseClassName
            == ExerciseClassName.ALTERNATING_ATTENTION)
        && currentTime > ExerciseController.this.maxTime) {
      ExerciseController.this.isTimeEnd = true;
      ExerciseController.this.numTurn =
            ExerciseController.this.numTurnTot;
      fillEmptyUserSolutions();
      checkUserSolution();
    }
  }
};
```

Codice 5.15: Definizione del metodo `handle` nella classe `ExerciseController`.

Per mostrare il tempo è stata quindi creata una progress bar, visibile in Figura 5.19, la cui classe è `ProgressTimeBar`. Tale classe estende `StackPane`, in quanto deve posizionare la stringa con il tempo sopra a una `ProgressBar`.

Il metodo `setProgressTimeBarValue`, in Codice 5.16, realizza l'aggiornamento dell'indicatore di progresso nella barra. Il valore impostato è compreso in un intervallo tra 0 e 1, dove lo 0 indica il progresso allo 0% (l'esercizio deve iniziare), mentre l'1 indica il progresso al 100% (l'esercizio è terminato).

```
public void setProgressTimeBarValue(long currentTime) {
  double currentValue = (double) currentTime / this.maxTime;
  this.bar.setProgress(currentValue);
  this.timeLabel.setText(TimeUtils
            .printTimeWithMills(currentTime));
  ...
}
```

Codice 5.16: Metodo per aggiornare l'indicatore di progresso nella barra del tempo.

Tale barra, potendo essere fonte di distrazione per il paziente che svolge l'esercizio, è stata nascosta e può essere mostrata solamente quando l'utente preme su un pulsante. Per rendere la scomparsa e la comparsa della barra più fluida, sono state aggiunte delle animazioni provenienti dalla libreria esterna AnimateFX.

PauseTransition

La classe `PauseTransition`, presente nella libreria di JavaFX, permette di creare una transizione, alla fine della quale, tramite il metodo `setOnFinished`, è possibile indicare quale tipo di operazione si deve svolgere.

Tale transizione è stata ampiamente utilizzata all'interno del progetto, per permettere la transizione dei singoli elementi all'interno della vista degli esercizi.

Un esempio di utilizzo di questa classe è mostrato nel Codice 5.17, dove la transizione del test viene impostata per 2 secondi, al termine dei quali si rimuove il test dal contenitore dove era stato posizionato e si avvia l'esercizio.

```
...
this.testDelay =
    new PauseTransition(javafx.util.Duration.millis(2000));
this.testDelay.setOnFinished(event -> {
  this.testPane.getChildren().clear();
  this.boxPane.getChildren().remove(this.testPane);
  this.testDelay = null;
  showCountdown();
});
this.testDelay.play();
...
```

Codice 5.17: Esempio di codice per settare un oggetto di tipo `PauseTransition`.

Le operazioni che è stato possibile realizzare tramite l'uso di `PauseTransition` sono:

- mostrare l'oggetto di test per un periodo prefissato, per poi rimuoverlo dalla vista;
- mostrare un'alternativa per un periodo di tempo e sostituirla con la successiva per un altro periodo di tempo;
- realizzare il countdown.

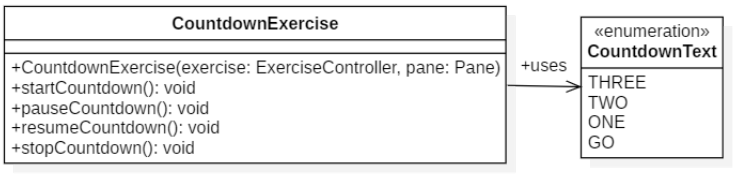


Figura 5.20: Diagramma delle classi del countdown.

CountdownExercise

Il conto alla rovescia, avviato prima di far partire il tempo, permette all'utente di prepararsi a svolgere il compito assegnato e a capire quando il tempo comincerà a scorrere.

Per realizzarlo è stata creata la classe `CountdownExercise`, in cui, tramite l'utilizzo della classe `PauseTransition`, è stato definito il funzionamento di tale oggetto.

Dal diagramma di Figura 5.20, è possibile vedere come tale classe includa metodi per mettere in pausa o fermare le transizioni, quando se ne ha la necessità.

Il metodo privato `nextValue` mostra il valore corrente del countdown, fa partire una transizione, al cui termine rimuove il valore mostrato e mostra il valore successivo (vedi Codice 5.18).

```
private void nextValue(CountdownText value) {
  this.textToShow.setText(BundleUtils
      .getBundle(value.getBundleCountdown()));
  this.pane.getChildren().add(this.textToShow);
  this.delayText.setOnFinished(event -> {
    this.pane.getChildren().remove(this.textToShow);
    this.delayEmpty.setOnFinished(event1 -> {
      if (value != CountdownText.GO) {
        nextValue(value.next());
      } else {
        this.exercise.showAlternatives();
      }
    });
    this.delayEmpty.play();
  });
  this.delayText.play();
}
```

Codice 5.18: Metodo per il funzionamento del countdown.

TurnStarsAnimation

La classe `TurnStarsAnimation` è stata creata per gestire il funzionamento della comparsa delle stelle all'interno dell'interfaccia grafica, che rappresentano il numero di turni e il turno corrente.

Tale classe imposta, all'interno di un contenitore, tante stelle vuote quanti sono il numero di turni dell'esercizio, poi, tramite il metodo `nextTurn`, mostrato nel Codice 5.19, va a sostituire la stella vuota con una stella piena nella posizione del numero del turno corrente.

```
public void nextTurn(HBox turnBox, int currentTurn) {
  StackPane nextTurnStar = getFullStar(currentTurn);
  new BounceIn(nextTurnStar).play();
  turnBox.getChildren().set(currentTurn, nextTurnStar);
}
```

Codice 5.19: Metodo per impostare la stella del turno successivo.

Come già successo per la barra del tempo, anche nel caso delle stelle sono state utilizzate le animazioni provenienti dalla libreria AnimateFX, per rendere più fluida la comparsa della nuova stella del turno.

5.4.3 Cattura degli eventi

La soluzione di molti esercizi, realizzati all'interno del framework, viene costruita attraverso la cattura dell'evento della pressione della barra spaziatrice.

In questo caso, è stato necessario aggiungere un listener, da registrare sullo stage primario dell'applicazione, per catturare l'evento.

Come visibile nel Codice 5.20, è stato definito un `EventHandler`, che cattura la pressione della barra spaziatrice e svolge delle operazioni (aggiunge alla soluzione dell'utente un valore booleano). Tale handler è, poi, aggiunto ai filtri degli eventi dello stage primario.

```
private void addSpaceListener() {
  this.spaceEventHandler = ((event) -> {
    if (event.getCode() == KeyCode.SPACE) {
      if (ExerciseController.this.exerciseClassName
              == ExerciseClassName.SART
          || ExerciseController.this.exerciseClassName
              == ExerciseClassName.LETTER_N_BACK
          || ExerciseController.this.exerciseClassName
              == ExerciseClassName.POSITION_N_BACK
          || ExerciseController.this.exerciseClassName
              == ExerciseClassName.STOP_SIGNAL) {
        if (ExerciseController.this.alternativesListTemp
            .size() != ExerciseController.this
```

```
                .previousLength) {
          int numObjectToAddToSolution =
            ExerciseController.this.alternativesListTemp
              .size() - (ExerciseController.this
              .previousLength != -1 ? ExerciseController.this
              .previousLength : 0) - 1;
          for (int i=0; i < numObjectToAddToSolution; i++) {
            ExerciseController.this.userSolutions.add(false);
          }
          ExerciseController.this.userSolutions.add(true);
          ExerciseController.this.previousLength =
            ExerciseController.this
              .alternativesListTemp.size();
        }
      }
      event.consume();
    }
  });
  this.mainClient.getPrimaryStage().getScene()
    .addEventFilter(KeyEvent.KEY_PRESSED,
          this.spaceEventHandler);
}
```

Codice 5.20: Metodo per aggiungere il listener allo stage primario.

Al termine dell'esercizio, tale listener deve essere rimosso dagli eventi dello stage primario, per evitare di generare eventi anomali, se la soluzione dell'utente prosegue la sua costruzione. Tale operazione viene svolta con il metodo mostrato nel Codice 5.21.

```
private void removeSpaceListener() {
  this.mainClient.getPrimaryStage().getScene()
    .addEventFilter(KeyEvent.KEY_PRESSED, eventKey -> {
    if (this.spaceEventHandler != null) {
      this.mainClient.getPrimaryStage().getScene()
        .removeEventFilter(KeyEvent.KEY_PRESSED,
            this.spaceEventHandler);
    }
    if (eventKey.getCode() == KeyCode.SPACE) {
      eventKey.consume();
    }
  });
  this.spaceEventHandler = null;
}
```

Codice 5.21: Metodo per rimuovere il listener dal primary stage.

5.4.4 Esercizi

Gli esercizi del framework, già precedentemente descritti nella sottosezione 3.6.1, sono stati implementati basandosi sulle classi definite in fase di progettazione.

Il funzionamento di ogni compito ha delle peculiarità, che lo differenzia da tutti gli altri e che ha richiesto delle implementazioni particolari. Tali particolarità verranno qui di seguito descritte con una breve panoramica.

Matrici attentive

La particolarità delle matrici attentive è che, ad ogni turno dello stesso esercizio, deve essere generata la stessa matrice in cui ricercare un certo stimolo.

Per poter realizzare questa matrice, al primo turno si generano gli stimoli all'interno della matrice in modo casuale, mentre, nei turni successivi, si deve dare in input al generatore di numeri random il seme utilizzato per generare la matrice al primo turno. Il seme è il valore del tempo corrente in millisecondi della prima generazione.

Gli stimoli generati devono essere posizionati all'interno di una matrice formata da 28 colonne e 12 righe. Per posizionare gli stimoli in modo distribuito all'interno della matrice, è stata realizzata una classe `AttentionalMatrixGrid`, la quale gestisce tutte le operazioni necessarie per costruire il `GridPane` con gli stimoli. Tale classe genera le posizioni in modo casuale e, per ogni posizione, genera in modo casuale anche l'allineamento dello stimolo (in alto, in basso, centrato, a destra o a sinistra). Il risultato grafico che è stato ottenuto, è mostrato in Figura 5.21.

SART

L'esercizio SART, invece, ha solamente la necessità di conoscere il tempo massimo assegnato per lo svolgimento dell'esercizio, per poter generare il numero corretto di stimoli da mostrare a video.

Il numero di stimoli è ottenuto dividendo il tempo massimo per il tempo totale necessario per mostrare un numero seguito da un simbolo.

La soluzione si ottiene confrontando i numeri random generati con il numero che fa da test. Il valore della soluzione è sempre true, a parte quando il numero generato è uguale al test, in quel caso è false.

La vista che si è ottenuta è visibile in Figura 5.22.

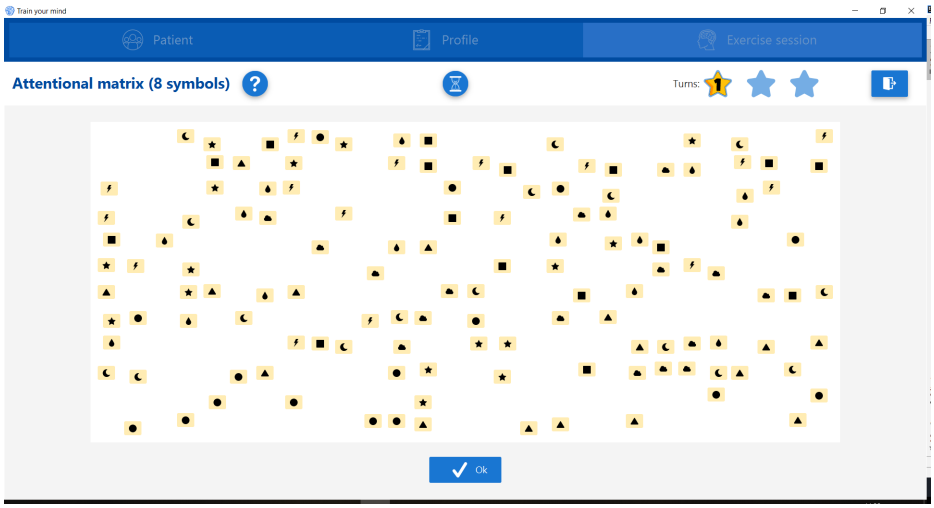
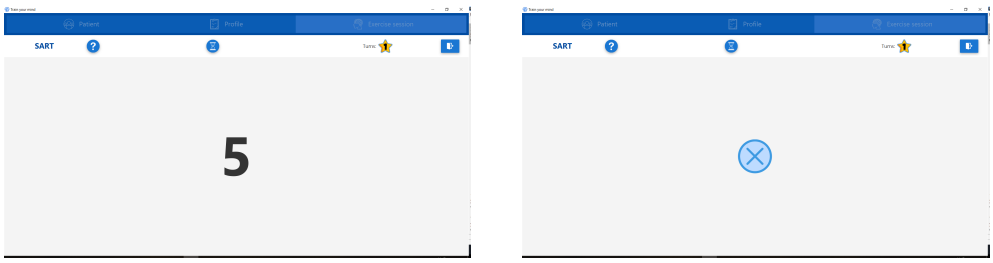


Figura 5.21: Screenshot della schermata dell'esercizio matrici attentive con 8 simboli.



(a) Alternativa mostrata. (b) Simbolo tra due alternative.

Figura 5.22: Screenshot delle schermate dell'esercizio SART, che mostra l'alternanza tra i numeri e i simboli.

Attenzione alternata

L'attenzione alternata deve generare delle figure geometriche in modo random.

Per poter costruire le figure geometriche è stata definita una classe statica `GeometricShape`, che contiene tutti i metodi per costruire le figure geometriche utilizzate all'interno del progetto. Alcune delle figure erano già disponibili all'interno della libreria di JavaFX, come ad esempio il cubo o il cerchio, mentre per altre figure è stato necessario costruirle con la classe `Polygon` (classe che permette di creare figure geometriche chiuse).

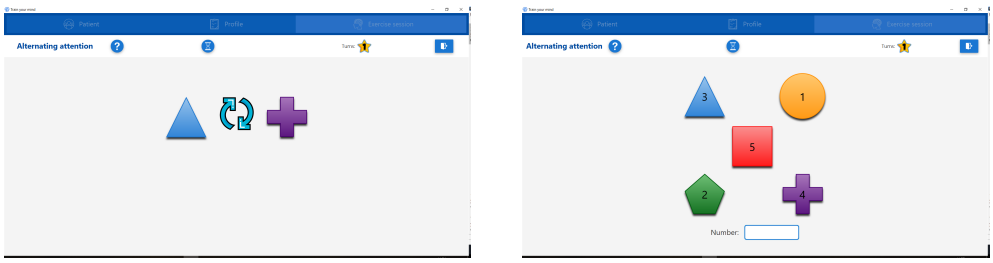
Come visibile nel Codice 5.22, per costruire una figura geometrica si deve creare un array di dati di tipo `Double`, il quale contiene le coordinate dei punti, che sono i vertici della figura. Tali punti devono, poi, essere aggiunti alla figura poligonale.

```
1  public static Polygon createTriangle(double height,
2                        String cssClassColor) {
3    Double[] points = new Double[]{
4        height / 2, 0.0,
5        height, height,
6        0.0, height
7    };
8    return createPolygon(points, cssClassColor);
9  }
10
11 private static Polygon createPolygon(Double[] points,
12                        String cssClassColor) {
13   Polygon polygon = new Polygon();
14   polygon.getPoints().addAll(points);
15   polygon.getStyleClass().add(CSS_SHAPE);
16   polygon.getStyleClass().add(cssClassColor);
17   return polygon;
18 }
```

Codice 5.22: Metodo che permette di creare un triangolo.

La classe dell'esercizio, `AlternatingAttention`, genera la posizione delle figure geometriche e i numeri assegnati alle figure geometriche, in modo casuale, e ottiene le figure geometriche grazie all'utilizzo del pattern Factory Method.

Il risultato che si è ottenuto, è visibile nella Figura 5.23. Nella classe `ExerciseController` è stato impostato sempre il focus sul campo in cui inserire il numero della figura e anche un listener per catturare l'evento di inserimento del numero. Il listener deve controllare che il valore inserito sia un numero e che tale valore sia compreso tra 1 e 5.



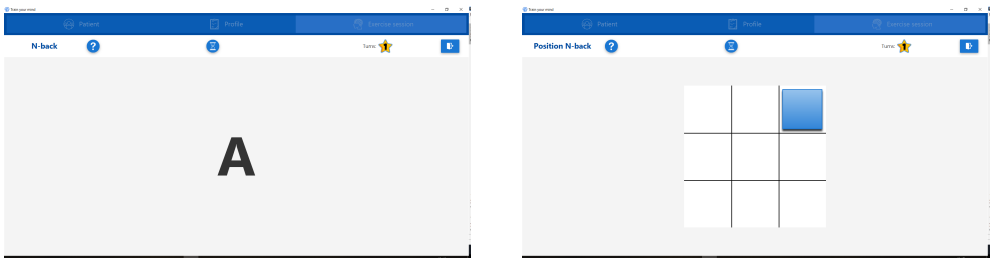
(a) Test. (b) Alternativa.

Figura 5.23: Screenshot delle schermate dell'esercizio attenzione alternata.

N-Back

L'esercizio N-Back, nelle sue due varianti "Letter N-Back" e "Position N-Back", deve generare delle alternative in cui devono comparire il 15% di stimoli uguali rispetto al numero totale di stimoli mostrati. Per ottenere questo funzionamento (Codice 5.23), si è prima individuato il numero di coppie di stimoli che devono comparire in base al numero di stimoli totali, poi si divide il numero di stimoli totali in intervalli. Il numero di intervalli serve per controllare se per quell'intervallo è già stata generata la coppia di stimoli, altrimenti la si genera alla fine, prima di passare all'intervallo successivo.

```
...
int totalNumberOfCouple = (int) (sizeList
        * this.getPercentEqualObject());
int interval = sizeList / totalNumberOfCouple;
int numCoupleGenerated = 0;
for (int i = 0; i < sizeList; i++) {
  if (((i + 1) % interval) == 0
        && ((i + 1) / interval) > numCoupleGenerated
        && numCoupleGenerated < totalNumberOfCouple) {
    this.alternatives.add(this.alternatives
        .get(this.alternatives.size() - 2
          * this.getValueOfN()));
    numCoupleGenerated++;
  } else {
    this.alternatives.add(getRandomLetter());
    if ((this.alternatives.size() - 2
            * this.getValueOfN() - 1) >= 0
        && this.alternatives.get(this.alternatives
            .size() - 1).equals(this.alternatives
            .get(this.alternatives.size() - 2
                * this.getValueOfN() - 1))) {
```



(a) Letter N-Back. (b) Position N-Back.

Figura 5.24: Screenshot delle schermate delle due varianti dell'esercizio N-Back.

```
22          numCoupleGenerated++;
23      }
24    }
25    ...
26  }
27  ...
```

Codice 5.23: Codice di `LetterNBack` che genera le alternative con un 15% di stimoli uguali.

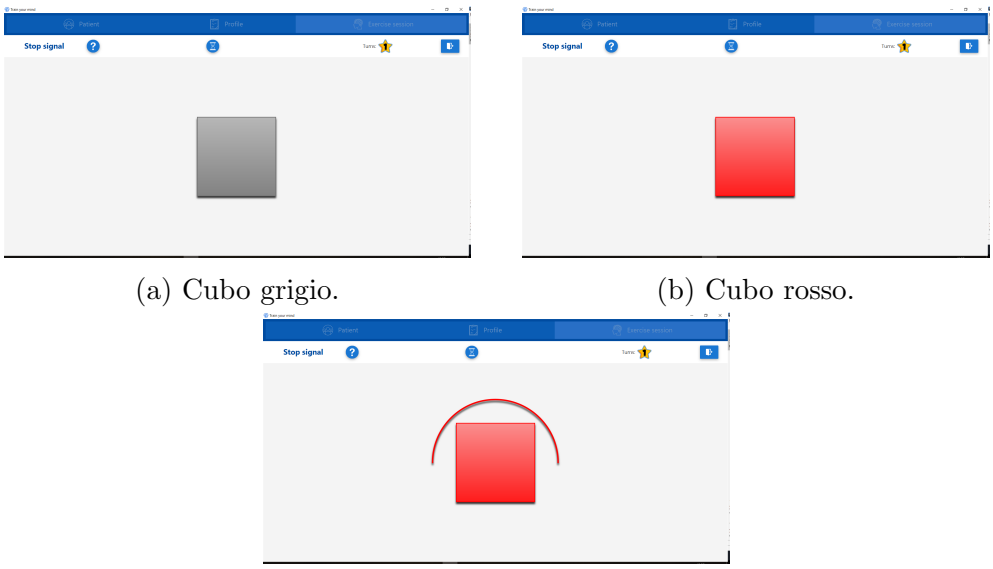
L'esercizio Letter N-Back, mostrato in Figura 5.24a, genera come alternative le 26 lettere dell'alfabeto, mentre Position N-Back, mostrato in Figura 5.24b, genera come alternative dei `GridPane` in cui posiziona un cubo in una posizione random.

Stop signal

Stop signal genera, in modo casuale, le tre tipologie di figure da mostrare: cubo grigio, cubo rosso e cubo rosso con arco.

La particolarità si verifica quando si genera la terza tipologia di figura, il cubo rosso con l'arco. Dato che il funzionamento dell'esercizio prevede che l'arco può comparire dopo un certo periodo di tempo, che dipende dal livello di difficoltà scelto per l'esercizio, alle soluzioni non sarà aggiunta una sola figura, ma due. La prima figura è il cubo rosso senza arco, la seconda è il cubo rosso sormontato dall'arco. Sarà compito della lista con i tempi di indicare quanto tempo dovrà essere visualizzata una figura e quanto l'altra. Invece, `ExerciseController` dovrà solo tenere conto della presenza di queste due figure per generare in modo corretto la soluzione data dall'utente.

Il risultato grafico, che si è ottenuto, è mostrato nelle immagini di Figura 5.25.



(a) Cubo grigio.

(b) Cubo rosso.

(c) Cubo rosso sormontato dall'arco.

Figura 5.25: Screenshot delle schermate dell'esercizio Stop signal.

Ordinamento crescente

L'esercizio di ordinamento crescente, la cui vista è mostrata in Figura 5.26, genera semplicemente sei numeri random, in un intervallo compreso tra 1 e un numero che dipende dal livello di difficoltà assegnato all'esercizio.

Il controller `ExerciseController` ha il compito di memorizzare l'ordine di selezione dei numeri fatta dall'utente e di segnalare all'utente quali sono i numeri che sono già stati selezionati.

Combina la figura al dettaglio

Nell'esercizio combina la figura al dettaglio, si generano le alternative selezionando, in modo casuale, tre delle figure disponibili. Dopo la generazione delle alternative, se ne sceglie una come soluzione e da questa si va a generare il test.

```
1 public static Image getPartOfImage(Class<?> clazz,
2                 String imagePath,
3                 int numPart) throws IOException {
4   BufferedImage bufferedImage = ImageIO
5         .read(clazz.getResourceAsStream(imagePath));
6   int dimensionOfPart;
7   if (bufferedImage.getWidth() > bufferedImage.getHeight()) {
```

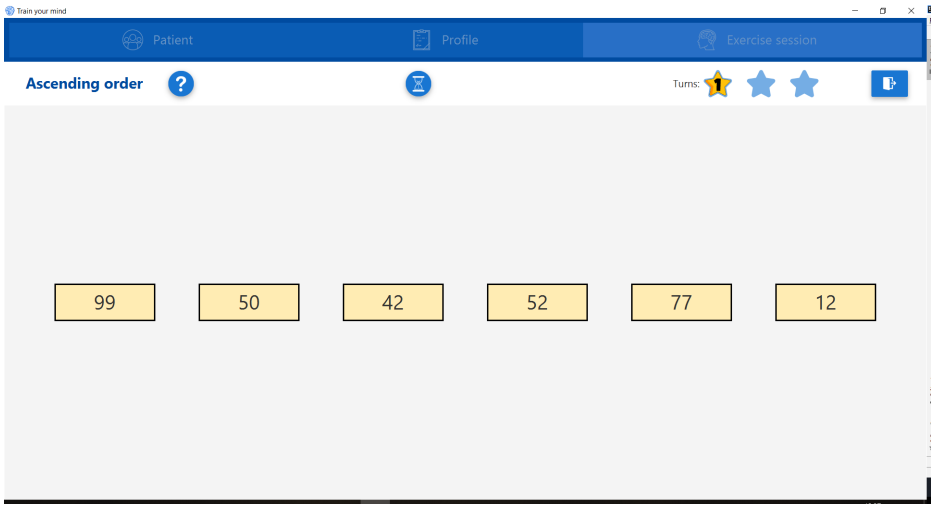


Figura 5.26: Screenshot della schermata dell'esercizio ordinamento crescente.

```
8       dimensionOfPart = bufferedImage.getHeight() / numPart;
9     } else {
10      dimensionOfPart = bufferedImage.getWidth() / numPart;
11    }
12    int x = RandomNumberUtils
13          .getRandomIntNumberFromZero(numPart - 1);
14    int y = RandomNumberUtils
15          .getRandomIntNumberFromZero(numPart - 1);
16    BufferedImage bufferedImagePart = bufferedImage
17        .getSubimage(x * dimensionOfPart, y * dimensionOfPart,
18              dimensionOfPart, dimensionOfPart);
19    return SwingFXUtils
20          .toFXImage(bufferedImagePart, null);
21      }
```

Codice 5.24: Metodo che genera una parte di immagine in modo casuale.

Il test è generato con il metodo di Codice 5.24. Come si può vedere, si legge l'immagine come una `BufferedImage`, cioè un'immagine di cui è disponibile un buffer di dati immagine. Si va, poi, a dividere l'altezza o la larghezza dell'immagine (si prende la dimensione con il valore più piccolo) per il numero di parti, in modo da ottenere la dimensione massima che deve avere ogni sezione dell'immagine. In seguito, si generano i valori di x e y in modo casuale. Questi due valori servono per indicare quale sezione dell'immagine è da prendere in considerazione, se quest'ultima fosse divisa in righe e colonne di un numero pari al numero di parti. A questo punto si genera la sezione dell'immagine grazie al metodo `getSubImage`. Tale metodo prende in input

la coordinata X, la coordinata Y, la larghezza e l'altezza della sezione che si deve ricavare dall'immagine.

Le operazioni svolte sono applicabili a una `BufferedImage` del package `java.awt.image`, ma dato che si sta utilizzando JavaFX, tale immagine deve essere convertita in una `Image` del package `javafx.scene.image`. Per fare questo, la libreria JavaFX mette a disposizione il metodo `toFXImage`, il quale copia i pixel della `BufferedImage` in un oggetto `Image` di JavaFX.

L'interfaccia grafica ottenuta per questo esercizio è stata mostrata in Figura 5.16.

Soluzione corretta dell'esercizio

Al termine del turno di ogni esercizio, dopo la verifica della soluzione ottenuta dall'utente, se la soluzione data è errata, è stato predisposto di mostrare la soluzione corretta dell'esercizio, con lo scopo di aiutare l'utente a capire i propri errori.

Il modo in cui le soluzioni corrette vengono proposte all'utente cambia in base all'esercizio, perché il tipo di dato mostrato è diverso. Nelle immagini di Figura 5.27 sono mostrate le seguenti soluzioni:

- Figura 5.27a: soluzione dell'esercizio delle matrici attentive, dove viene riproposta la stessa matrice su cui il paziente ha svolto l'esercizio, con gli stimoli nelle stesse posizioni. Lo stimolo è colorato di verde quando fa parte della soluzione corretta, mentre è colorato di rosso se l'utente ha dato per quello stimolo la risposta errata.

- Figura 5.27b: soluzione dell'esercizio SART, dove per ogni numero presentato si indica una spunta verde per la risposta data corretta, mentre una ics rossa per la risposta data sbagliata.

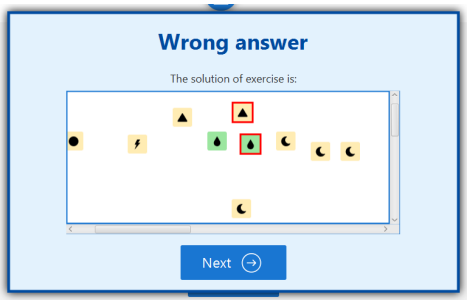
- Figura 5.27c: soluzione dell'esercizio di attenzione alternata, dove per ogni figura geometrica si indicano i numeri ad essa associata, i quali saranno di colore rosso se l'utente non ha indicato quel numero.

- Figura 5.27d: soluzione dell'esercizio Letter N-Back, che restituisce la soluzione come il SART.

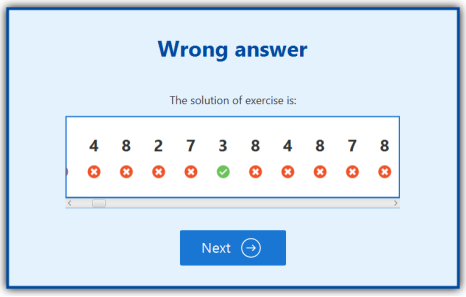
- Figura 5.27e: soluzione dell'esercizio Position N-Back, che restituisce la soluzione come il precedente.

- Figura 5.27f: soluzione dell'esercizio Stop signal, che restituisce la soluzione come l'esercizio precedente.

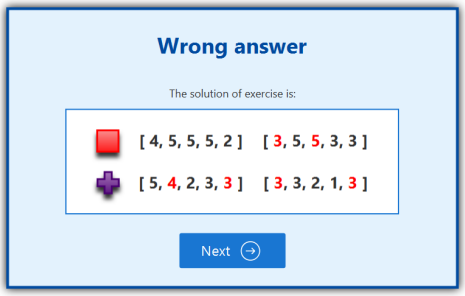
- Figura 5.27g: soluzione dell'esercizio di ordinare i numeri in modo crescente, in cui si mostra l'ordinamento corretto dei numeri, i quali vengono indicati in rosso se sono stati posizionati nella posizione sbagliata della sequenza.
- Figura 5.27h: soluzione dell'esercizio per combinare la figura al dettaglio proposto, dove si mostra semplicemente la figura completa a cui il dettaglio apparteneva.



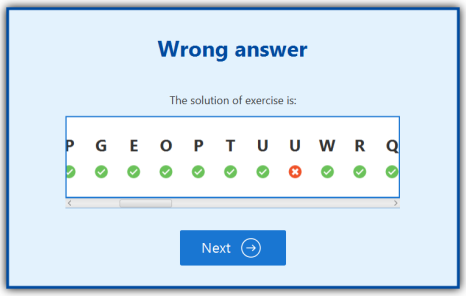
(a) Matrici attentive.



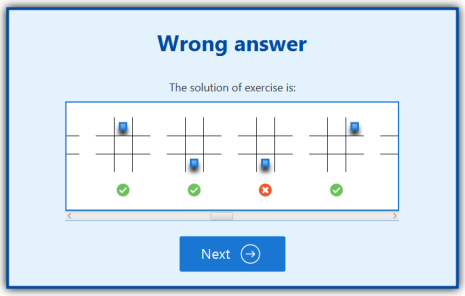
(b) SART.



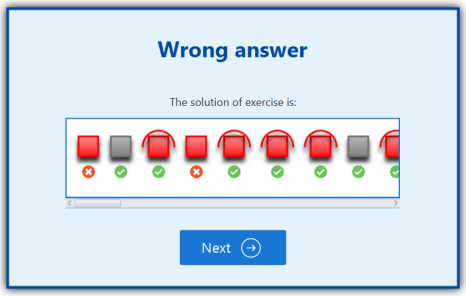
(c) Attenzione alternata.



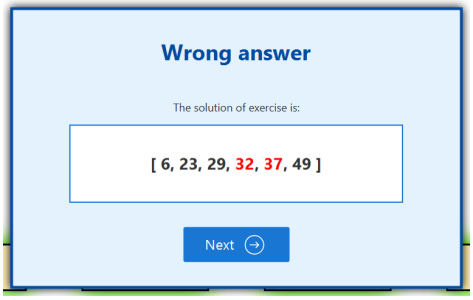
(d) Letter N-Back.



(e) Position N-Back.



(f) Stop signal.



(g) Ordinamento crescente.

(h) Combina la figura al dettaglio.

Figura 5.27: Screenshot delle schermate che mostrano la soluzione corretta dell'esercizio.

Capitolo 6

Sperimentazione

La progettazione e la realizzazione di un software possono portare ad ottenere idealmente un buon progetto, ma non è detto che, nella pratica, tale software sia effettivamente utile come si pensava.

Per mostrare l'effettiva applicabilità del progetto "Train your mind" per l'allenamento cognitivo di pazienti sani e malati, è stata avviata una sua sperimentazione. Il software è stato dato ai medici del Centro Studi e Ricerche in Neuroscienze Cognitive di Cesena, i quali hanno svolto qualche sessione di allenamento a un campione di pazienti malati e un campione di individui sani. Lo scopo di tale sperimentazione è quello di vedere come gli utenti si approcciano al software, ricevere qualche feedback riguardo alle funzionalità presenti e capire se, dai dati ricavati dalle sessioni di allenamento, il software permette di distinguere i pazienti sani da quelli malati.

Qui di seguito verrà presentato il campione di paziente con cui è stato testato il progetto e i dati ricavati dalla sperimentazione.

6.1 Campione

Per svolgere la sperimentazione è stato necessario, in primis, selezionare il campione su cui si sarebbero svolte le sessioni di allenamento.

I pazienti sono stati selezionati tra gli individui che frequentano di norma il Centro di Neuroscienze. In questo modo, si sono potute svolgere le sedute direttamente all'interno della struttura. Il campione che si è andato a costituire è formato da undici soggetti, sei malati e cinque sani, di età compresa tra i 33 e i 77 anni.

I pazienti malati selezionati riportano tutti delle lesioni riguardanti il lobo frontale o temporale del cervello. La posizione di queste lesioni provoca nei pazienti alterazioni nella memoria o nello svolgimento delle funzioni esecutive.

La causa di queste lesioni è stata provocata da patologie differenti, tra le quali troviamo:

- la neoplasia, forma tumorale che prevede la crescita eccessiva di tessuto rispetto al normale;
- l'aneurisma cerebrale, malformazione vascolare che forma sacche di sangue, che se si rompono possono andare ad irrorare una zona del cervello, uccidendo così i neuroni presenti;
- il cavernoma cerebrale, forma tumorale benigna che consiste in un agglomerato di vasi sanguigni, che se si rompono possono provocare gli stessi effetti dell'aneurisma;
- l'ictus, attacco cerebrale che provoca la morte delle cellule cerebrali. La morte può essere provocata da una mancanza di flusso di sangue oppure da un sanguinamento.

Per ogni soggetto del campione sono stati raccolti i dati principali, come l'età, la scolarità, il tipo di lesione e l'eziologia, in modo da poter fare raffronti con i dati raccolti. Il riassunto delle informazioni riguardanti il campione è visibile nella Tabella 6.1.

Un particolare da tenere in considerazione quando si andranno ad analizzare i risultati ottenuti, è che quattro di questi soggetti (quelli più anziani) non sono tecnologicamente esperti, perciò è stato necessario insegnare loro ad usare il mouse o il touch pad. In quasi tutti i casi, dopo aver appreso le informazioni basilari, i soggetti sono riusciti a svolgere i compiti in autonomia, ad eccezione di un soggetto, il quale ha avuto la necessità di essere sostituito dal medico, quando era necessario lo svolgimento di operazioni con il mouse.

6.2 Metodologia

La sperimentazione svolta ha previsto l'esecuzione di due sessioni di allenamento per ogni soggetto del campione. Durante queste sessioni sono stati svolti sempre gli stessi esercizi, ma con due livelli di difficoltà differenti: la prima sessione è stata svolta con il livello di difficoltà "principiante", mentre la seconda con quello "intermedio".

Dal catalogo di esercizi disponibili sul framework, sono stati selezionati sette esercizi. Questi, però, non sono stati eseguiti sempre nello stesso ordine per ogni soggetto campione. Anzi, in alcuni casi, è stato necessario ripetere alcuni esercizi, perché il soggetto non aveva capito il compito da svolgere o

Tabella 6.1: Dati del campione.

Paziente	**Età**	**Scolarità**	**Lesione**	**Eziologia**
M. O.	37	13	Frontale sinistro	Neoplasia frontale sinistro
V. G.	60	8	Frontale-ventromediale	Aneurisma della comunicazione anteriore
C. M. L.	43	11	Frontale destro	Cavernoma frontale destro
B. S.	67	8	Frontale sinistro	Ictus cerebri frontale sinistro
E. G.	55	19	Temporo-parietale destro	Ictus cerebri emisferico destro
M. C.	67	5	Frontale bilaterale	Ictus cerebri
A. B.	51	13		
B. S.	33	19		
C. M.	47	8		
L. G.	64	8		
P. N.	77	8		

Nome esercizio	**Funzionalità allenata**	**Tempo max**
Matrici attentive con 8 figure	Attenzione selettiva	10 min
SART	Attenzione sostenuta	10 min
Attenzione alternata	Attenzione complessa	5 min
Letter N-back	Memoria lavoro	3 min
Position N-back	Memoria lavoro	3 min
Stop signal	Attenzione sostenuta	3 min
Ordinamento crescente	Attenzione complessa	3 min

Tabella 6.2: Informazioni sugli esercizi svolti.

non stava prestando attenzione. Gli esercizi selezionati sono elencati nella Tabella 6.2.

Le sessioni di allenamento dei singoli soggetti sono state svolte, nella maggior parte dei casi, in giornate differenti. Si sono verificate situazioni in cui i soggetti non erano disponibili per più giorni, perciò è stato necessario condensare le sessioni una di seguito all'altra, anche se questo potrebbe aver inficiato le prestazioni del soggetto.

Ogni sessione ha richiesto circa un'ora per essere completata e in questa occasione sono stati raccolti i tempi impiegati per concludere ogni singolo esercizio e il numero di errori e risposte corrette date.

6.3 Risultati

I dati raccolti attraverso lo svolgimento delle sessioni di esercitazione sono stati analizzati per vedere se era possibile trovare una qualche differenza nelle prestazioni tra soggetti di categorie, età o scolarità differenti. Lo scopo dell'analisi dei risultati è quella di capire se il software è in grado di distinguere i soggetti sani da quelli malati.

Prima di procedere con l'analisi dei dati, bisogna tenere conto che circa il 10% dei dati sono mancanti. Il motivo è da ricercare tra i seguenti fenomeni:

- per un soggetto non è stata eseguita la seconda sessione con il livello di difficoltà "principiante";
- il risultato restituito dal software al termine di un esercizio non era esatto, perciò l'errore è stato corretto per svolgere tutte le altre sessioni;
- i compiti Letter N-back e Position N-back, con livello di difficoltà "intermedio", non sono stati spiegati correttamente dal medico al campione

di pazienti sani. Questo ha provocato il verificarsi di errori, che non sarebbero sorti, se non ci fosse stato questo problema.

6.3.1 Analisi in base al tempo

Per parte dei risultati ottenuti, può essere utile fare una valutazione in base al tempo impiegato dal soggetto per concludere l'esercizio.

In questo caso, si analizzano i dati provenienti dagli esercizi di matrici attentive, attenzione alternata e ordinamento crescente, in quanto gli altri esercizi proposti prevedono lo svolgimento del compito durante tutto il tempo assegnato.

Confrontando il tempo impiegato dai soggetti malati e da quelli sani, attraverso i grafici di Figura 6.1 e 6.2, è possibile notare come il software sia in grado di distinguere le due categorie. Infatti, i soggetti sani hanno riflessi più veloci, che gli permettono di svolgere i compiti assegnati in un tempo minore rispetto a quelli malati.

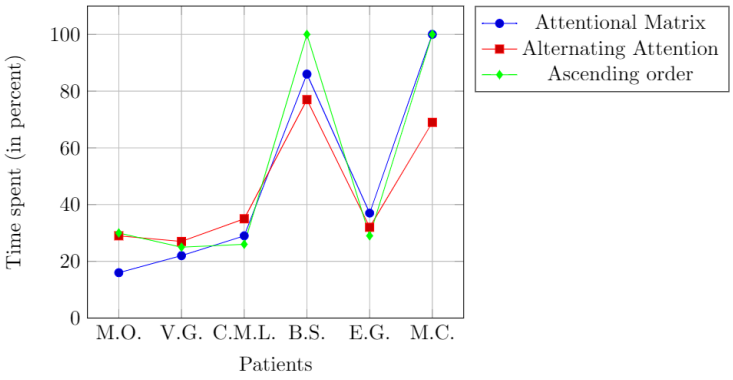
I picchi individuati nei grafici corrispondono a quei soggetti non tecnologicamente esperti. Questo fattore va ad incidere pesantemente sulle prestazioni del soggetto. Si può anche notare, però, come questi individui, abbiano avuto un leggero miglioramento nelle prestazioni durante la seconda sessione di allenamento, anche se è stata svolta con un livello di difficoltà superiore. Il motivo lo si può individuare nel fatto che i soggetti avevano preso più dimestichezza con la tecnologia e si sentivano più sicuri nell'operare in autonomia.

Il miglioramento tra la prima e la seconda sessione si nota anche in generale un po' per tutti i soggetti. Questo è dovuto al fatto che nella prima sessione, i pazienti si approcciavano con il compito per la prima volta e dovevano ancora capirne a pieno il funzionamento. Nella seconda sessione, invece, il paziente aveva già svolto l'esercizio, perciò si sentiva più sicuro di sé e ha proceduto più velocemente.

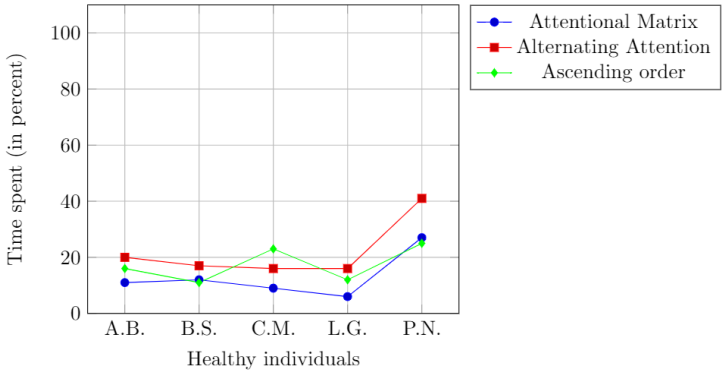
6.3.2 Analisi in base agli errori

Un'altra analisi, che si può svolgere sui risultati, è quella che prende in considerazione la percentuale di errori commessi dai soggetti per ogni singolo esercizio. Il numero di errori è stato ottenuto mettendo insieme i valori delle omissioni e i valori dei falsi allarmi.

Anche in questo caso, come già verificatosi analizzando le prestazioni in base al tempo, il software è in grado di distinguere i soggetti sani da quelli malati. Guardando i grafici di Figura 6.3 e 6.4, si nota che la percentuale degli errori commessi è più alta nei soggetti malati, rispetto a quelli sani.

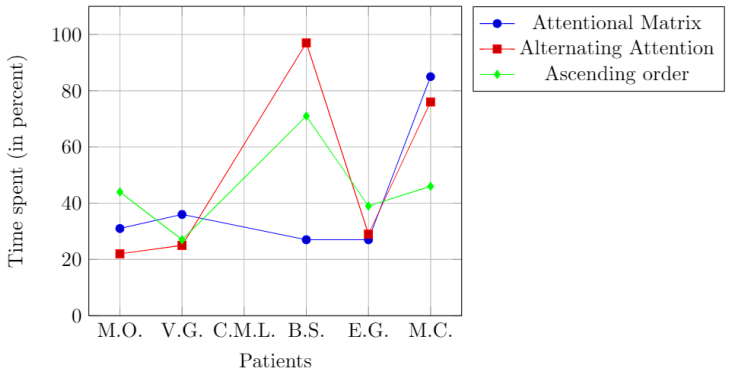


(a) Grafico delle prestazioni dei pazienti.

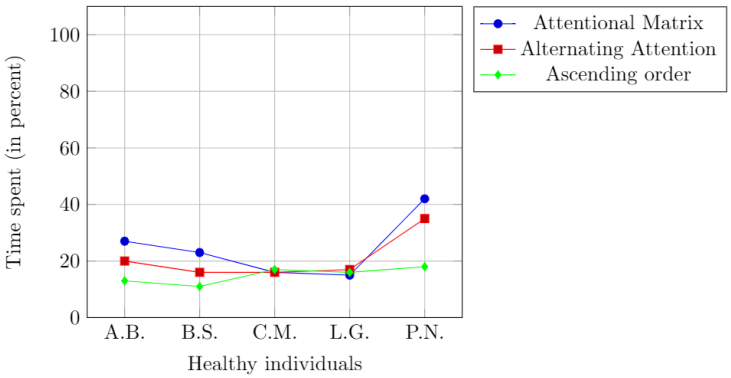


(b) Grafico delle prestazioni degli individui sani.

Figura 6.1: Grafici delle prestazioni dei soggetti campione in termini di percentuale di tempo impiegato per terminare l'esercizio con il livello principiante.

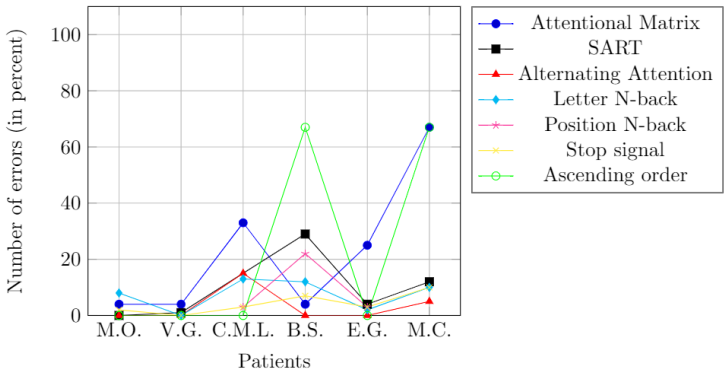


(a) Grafico delle prestazioni dei pazienti.

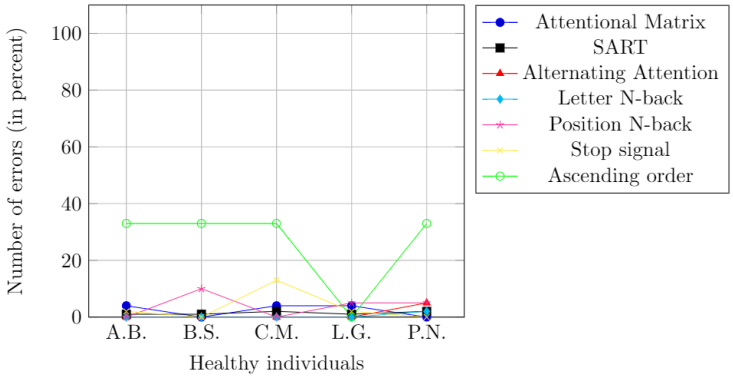


(b) Grafico delle prestazioni degli individui sani.

Figura 6.2: Grafici delle prestazioni dei soggetti campione in termini di percentuale di tempo impiegato per terminare l'esercizio con il livello intermedio.



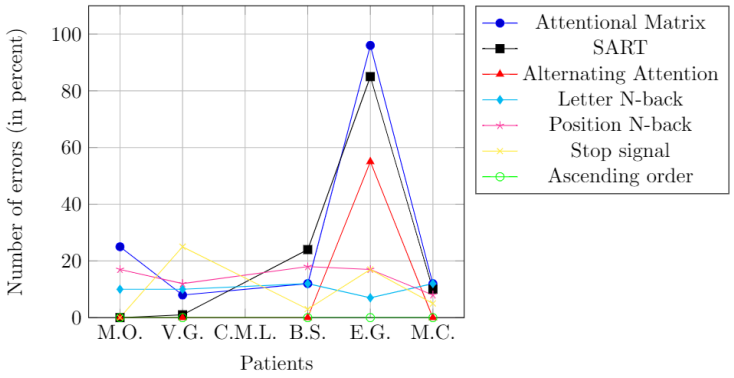
(a) Grafico delle prestazioni dei pazienti.



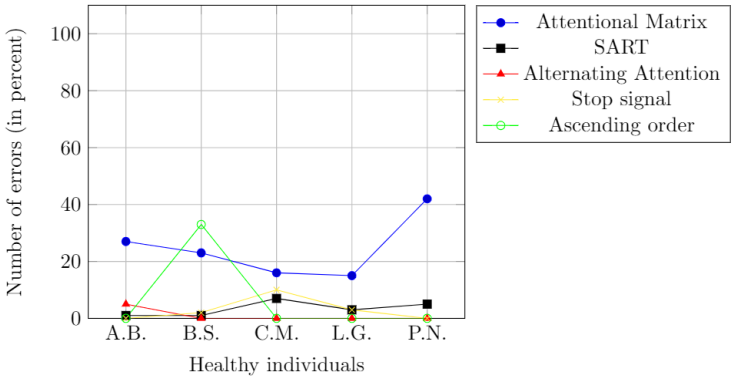
(b) Grafico delle prestazioni degli individui sani.

Figura 6.3: Grafici delle prestazioni dei soggetti campione in termini di percentuale di errori commessi durante lo svolgimento del compito con il livello principiante.

Un'anomalia si verifica nell'esercizio di ordinamento crescente, in cui gli individui sani sembra che commettano maggiormente degli errori rispetto ai pazienti malati (vedi Figura 6.3b). La causa che ha portato a questa anomalia è dovuta al fatto che il soggetto può aver dato poca importanza al compito, non prestandogli la giusta attenzione, e nel tentativo di svolgerlo nel minor tempo possibile, ha commesso più errori. Questo è, perciò, l'unico test a non discriminare tra le due tipologie di campioni in base alla percentuale di errori commessi.



(a) Grafico delle prestazioni dei pazienti.



(b) Grafico delle prestazioni degli individui sani.

Figura 6.4: Grafici delle prestazioni dei soggetti campione in termini di percentuale di errori commessi durante lo svolgimento del compito con il livello intermedio.

6.4 Discussione

Questa sperimentazione ha dimostrato che "Train your mind" è un software che aiuta e facilita lo svolgimento delle sessioni di allenamento con i pazienti.

I soggetti, che hanno partecipato a queste sedute, hanno trovato il software facile da usare e non hanno sentito il peso della difficoltà degli esercizi, anche se alcuni di questi richiedevano molta concentrazione.

Analizzando i risultati ottenuti, si può affermare che il sistema è in grado di distinguere i pazienti sani da quelli malati. Ciò è possibile farlo esaminando le prestazioni che hanno i differenti soggetti in termini di tempo impiegato e di errori commessi.

Inoltre, sono stati individuati due fattori, che vanno molto ad incidere sulle prestazioni dei soggetti: l'età e la conoscenza della tecnologia. L'età va sicuramente ad intaccare la reattività del soggetto, ma la scarsa, se non assente, conoscenza tecnologica fa aumentare drasticamente il tempo necessario per terminare un compito. In molti casi, il verificarsi di una condizione implica anche il verificarsi dell'altra.

Il livello di istruzione, nella maggior parte dei casi, non sembra avere una qualche correlazione con le prestazioni ottenute. L'unico problema riscontrato in questo caso è nello svolgimento di Letter N-back. I soggetti con anni di scolarità minori hanno avuto difficoltà nel riconoscere le lettere dell'alfabeto straniere e questo ha causato perdita di concentrazione e impossibilità nel soggetto di confrontare tali lettere con le altre mostrate.

Nel futuro dovranno essere svolti ulteriori test per verificare se l'allenamento cognitivo erogato con questo software è effettivamente efficace per il miglioramento delle principali funzioni cognitive.

Capitolo 7

Conclusioni

Il lavoro svolto in questa tesi aveva l'obiettivo di realizzare un software applicativo che consentisse lo svolgimento di un allenamento cognitivo su soggetti anziani, durante l'invecchiamento sano o patologico. Questo strumento doveva affiancare l'operato del medico, semplificando lo svolgimento delle sessioni di esercitazione e andando ad allenare specifiche aree cognitive.

I requisiti, definiti in fase di analisi, sono stati tutti soddisfatti, permettendo di ottenere così un software conforme alle necessità dei futuri utenti. Inoltre, grazie all'utilizzo di strumenti e tecnologie noti e standard, si è reso possibile l'estensione e la modifica del sistema per eventuali sviluppi futuri.

L'interfaccia grafica del software è stata realizzata tramite l'utilizzo del software applicativo JavaFX. Sfruttando le sue potenzialità, è stato possibile separare la logica comportamentale dall'aspetto grafico, ottenendo un'interfaccia user-friendly, ispirata al mondo del web.

Il cuore pulsante di tutto il progetto, però, è stata la definizione e la realizzazione del framework degli esercizi. Questo framework permette allo sviluppatore di implementare il proprio esercizio e di aggiungerlo in modo semplice, per poterlo poi utilizzare all'interno una sessione di esercitazione. Il framework è anche stato corredato da sette tipologie di esercizi, che permettono di allenare specifiche aree cognitive e la memoria lavoro di un soggetto.

Attraverso la sperimentazione svolta su un campione di soggetti, è stato possibile vedere come gli utenti finali interagiscono con il software sviluppato e ricevere dei feedback per eventuali miglioramenti futuri. Grazie ai dati raccolti da questa sperimentazione è stato possibile verificare che il sistema è in grado di permettere di fare una distinzione tra pazienti sani e pazienti malati, solamente andando a confrontare e analizzare i risultati ottenuti da questi negli esercizi.

Il progetto, però, non è da considerarsi concluso, in quanto le funzionalità sviluppate sono solamente il mattone fondante delle potenzialità che può avere. Già arrivati a questa fase si possono individuare sviluppi futuri, come ad esempio la realizzazione di una domiciliazione che permette al medico di assegnare direttamente un'esercitazione al paziente dalla propria postazione, senza dover configurare manualmente scheda clinica e del profilo nel software dato in dotazione al paziente. Un'altra idea può essere quella di permettere una maggiore personalizzazione del look and feel dell'interfaccia grafica, in modo da rendere il software accessibile. Inoltre, si può prevedere un ampliamento della sezione con la reportistica, per permettere una migliore analisi dei dati raccolti durante le sessioni, da parte del medico.

Quindi, possiamo dire, in conclusione che il lavoro di tesi svolto ha portato alla realizzazione di un sistema che è già in grado di essere distribuito al pubblico, perché mette a disposizione le funzionalità base e un primo set di esercizi da svolgere. Grazie ai feedback ricevuti dagli utenti finali, possiamo anche affermare che risulta facile da utilizzare, rendendo anche semplice l'esecuzione del compito da svolgere. Nel futuro, dovrà essere testata l'efficacia di questo strumento per il miglioramento delle funzioni cognitive dei soggetti su cui viene applicato.

Ringraziamenti

Arrivati al termine di questo percorso vorrei ringraziare tutti coloro, che mi hanno sostenuto, aiutato e incoraggiato ad andare avanti per raggiungere questo traguardo.

In primo luogo, ringrazio il professor Mirko Viroli per avermi proposto e concesso di lavorare a questo progetto, che avrà sicuramente un'utilità in campo medico. Lo ringrazio, soprattutto, per la sua disponibilità, per l'aiuto e i consigli ricevuti durante il corso di questa esperienza.

Un grazie va anche al professor Alessio Avenanti, che mi ha accompagnato in questa esperienza, facendomi scoprire un po' il mondo delle neuroscienze. Insieme a lui, ringrazio anche la professoressa Caterina Bertini e il dottor Davide Braghittoni, insieme a tutti gli specializzandi e i ricercatori del Centro Studi e Ricerche in Neuroscienze di Cesena, che mi hanno affiancato nella definizione dei compiti di allenamento da realizzare e hanno svolto sui loro pazienti alcune sessioni di allenamento con questo software. Ringrazio, inoltre, Francesco Tortora che si è occupato di svolgere i test sui pazienti sani.

Ringrazio la professoressa Silvia Mirri che, dopo aver preso visione del mio progetto, mi ha dato ottimi consigli e nuovi spunti per migliorarne l'interfaccia grafica.

Un grazie particolare va ai miei genitori, che mi hanno sempre sostenuto e che, senza i quali, non sarei mai arrivata a raggiungere questa meta.

Per ultimi, ma non per importanza, ringrazio i miei amici che mi sono stati accanto sia nei momenti belli, che in quelli più difficili.

Bibliografia

[1] Gul A. Agha. *Actors: A model of concurrent computation in distributed systems*. Report. MIT Artificial Intelligence Laboratory, 1985.

[2] *Akka documentation*. Ver. 2.5.23. 2019. URL: https://doc.akka.io/docs/akka/current/index.html.

[3] Denise Albani et al. *Morbo di Alzheimer: caratteristiche, cause , diagnosi e fasi della malattia*. 2016. URL: https://www.medicina360.com/Morbo_di_alzheimer.html.

[4] Deepak Alur, John Crupi e Dan Malks. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall, 2003.

[5] Riccardo Borgacci et al. *Corpi di Lewy*. 2018. URL: https://www.my-personaltrainer.it/salute-benessere/corpi-di-lewy.html.

[6] Riccardo Borgacci et al. *Demenza frontotemporale*. 2018. URL: https://www.my-personaltrainer.it/salute-benessere/demenza-frontotemporale.html.

[7] Massimo Canorro. *Malattie neurodegenerative: le principali patologie*. 2017. URL: https://www.paginemediche.it/benessere/corpo-e-mente/malattie-neurodegenerative-le-principali-patologie.

[8] Scott Chacon e Ben Straub. *Pro Git*. Apress, 2019.

[9] *Derby Reference Manual*. Ver. 10.15. The Apache DB Project. 2019. URL: https://db.apache.org/derby/docs/10.15/ref/index.html.

[10] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[11] Serge Gauthier et al. «Mild cognitive impairment». In: *The Lancet* 367.9518 (2006), pp. 1262–1270.

[12] *Gradle User Manual*. Ver. 5.5. 2019. URL: https://docs.gradle.org/current/userguide/userguide.pdf.

[13] Antonio Griguolo. *Demenza Vascolare*. 2019. URL: https://www.my-personaltrainer.it/salute-benessere/demenza-vascolare.html.

[14] C. Herrera et al. «Positive effects of computer-based cognitive training in adults with mild cognitive impairment». In: *Nuropsychologia* 50 (2012), pp. 1817–1881.

[15] Nicole T.M. Hill et al. «Computerized Cognitive Training in Older Adults With Mild Cognitive Impairment or Dementia: A Systematic Review and Meta-Analysis». In: *Am J Psychiatry* 174.4 (apr. 2017), pp. 329–340.

[16] Alexandra M. Kueider et al. «Computerized Cognitive Training with Older Adults: A Systematic Review». In: *Plos One* 7 (lug. 2012). A cura di Sonia Brucki (University of São Paulo Brazil).

[17] *Malattia di Huntington: come potrebbero aiutare le cellule staminali?* 2016. URL: https://www.eurostemcell.org/it/malattia-di-huntington-come-potrebbero-aiutare-le-cellule-staminali.

[18] *MySQL 8.0 Reference Manual*. Ver. 8.0. Oracle Corporation. 2019. URL: https://dev.mysql.com/doc/refman/8.0/en/.

[19] *Rotary*. URL: https://www.rotary.org/.

[20] Herbert Schildt. *Java. The Complete Reference. IX Edition*. McGraw-Hill Education, 2014.

[21] The Webweavers. *Il morbo di Parkinson*. 2014. URL: http://www.scienzagiovane.unibo.it/malattie-cervello/3-Parkinson.html.

[22] The Webweavers. *La Sclerosi Laterale Amiotrofica (SLA)*. 2014. URL: http://www.scienzagiovane.unibo.it/malattie-cervello/4-SLA.html.