

Sistema informatico interattivo per la
stimolazione e la riabilitazione cognitiva
del paziente con malattia di Alzheimer a
domicilio
INFORMA

Descrizione del sistema

Sommario

1	Descrizione generale	4
1.1	Scopo del software	4
1.2	Attori	4
1.2.1	Terapisti	4
1.2.2	Pazienti	4
1.2.3	Amministratore di team clinico	4
1.3	Componenti	4
1.4	Configurabilità del sistema	5
2	DB	6
2.1	Tabelle relative alle utenze	6
2.2	Tabelle relative alle attività	7
2.3	Tabelle delle entità della libreria	8
2.4	Altre tabelle	9
3	API	10
3.1	Server e App	10
3.2	Middlewares:	10
3.3	Routes	10
3.3.1	Elenco completo delle routes api	11
3.4	Moduli	12
3.5	Preparazione eseguibili	12
4	Clients	14
4.1	CMS	14
4.1.1	Moduli principali	14
4.1.1.1	Core	14
4.1.1.2	Communication	15
4.1.1.3	Calendar	15
4.1.1.4	Interchange	15
4.1.1.5	Library	15
4.1.1.6	Patient	15
4.1.1.7	Performance	15
4.1.1.8	Universal	16
4.1.2	Preparazione eseguibili	16
4.2	APP	17
4.2.1	Comunicazione con API	17

4.2.2	Moduli principali	19
4.2.2.1	Core	19
4.2.2.2	Communication	19
4.2.2.3	Training	19
4.2.2.4	Universal	19
4.2.3	Preparazione eseguibili	19
4.3	UNIVERSAL	21
4.3.1	Moduli principali	21
4.3.1.1	Tool	21
4.3.1.2	Session	21
4.3.1.3	Updater	21
4.3.1.4	Visibility	21
4.3.1.5	Auth	21
4.3.1.6	Download	21
4.3.1.7	Product	22
4.3.1.8	User	22
4.3.1.9	Sync	22
4.3.1.10	Platform	22
4.3.1.11	Navigation	22
5	Attività	23
6	Allegato A	24

1 Descrizione generale

1.1 Scopo del software

INFORMA nasce come piattaforma atta all'esercitazione neuro-cognitiva volta al trattamento della patologia Alzheimer, il sistema si configura come un veicolo di attività dedicate a dispositivi touchscreen portatili i quali contenuti siano configurabili da un team di terapisti che segue diversi pazienti.

Uno dei requisiti fondamentali è la possibilità di offrire una configurabilità avanzata delle esercitazioni in modo da poter offrire un allenamento neuro-cognitivo studiato specificatamente per il singolo paziente.

1.2 Attori

Il Sistema INFORMA verrà utilizzato da 3 tipologie di utenti se si escludono eventuali sviluppatori dedicati all'ampliamento dei suoi contenuti.

1.2.1 Terapisti

Questi utenti avranno accesso ad una piattaforma web per configurare le esercitazioni da fornire ai pazienti.

1.2.2 Pazienti

Utenti affetti da patologia Alzheimer che accederà ad attività a loro proposte sotto indicazione di un terapeuta

1.2.3 Amministratore di team clinico

L'amministratore del team clinico sarà un terapeuta con facoltà di gestire le utenze terapisti nonché visibilità su tutti i pazienti gestiti da proprio team di terapisti

1.3 Componenti

Il Sistema INFORMA ha un web server Api (da qui in avanti riferito semplicemente come "API") che insiste su un Database (occasionalmente riferito come "DB" da qui in avanti) attualmente implementato in MYSQL ma non strettamente legato a tale infrastruttura.

Il team clinico utilizza un client web che utilizza quanto fornito da API per modificare la libreria di temi e stimoli, creare esercitazioni e gestire gli utenti e la programmazione delle attività (tale client sarà da qui in avanti riferito come "CMS").

Il paziente ha a disposizione un client compilato per supporti mobile che permette di accedere alle attività programmate ed eseguirle secondo le indicazioni del CMS (tale client paziente sarà da qui in avanti riferito come "APP")

Scopo del sistema è creare delle esercitazioni specifiche (da qui in poi definite come "ripetizioni") in una serie di attività neuro cognitive (da qui in seguito "attività") a partire da una serie di contenuti media (da qui in poi definiti come "stimoli") e pianificarle nel tempo in sessioni dedicate (da qui in avanti riferite come "esercitazioni") per ogni paziente secondo le necessità dello stesso e modulare tale pianificazione a secondo i dati di performance raccolti.

1.4 Configurabilità del sistema

Il sistema INFORMA nasce con lo scopo di creare una piattaforma indipendente dalle effettive attività in essa contenute e dai contenuti da essi utilizzati.

L'indipendenza sopra citata viene implementata grazie a tre livelli di configurazione, uno per le attività da veicolare, uno per gli stimoli da esse utilizzate e uno per le effettive esercitazioni configurate per ognuna delle attività (da qui in avanti riferite come "ripetizioni").

Le attività da veicolare sono configurate direttamente da Database dove vengono definiti i parametri di configurazione che accettano, nonché i parametri di performance che producono insieme ad informazioni di servizio come la loro posizione per il download tali informazioni sono poi passate tramite API all'APP per il loro lancio e raccolta dei dati di performance e a CMS per la presentazione dei dati di performance.

La configurazione degli stimoli e ripetizioni avviene direttamente da CMS ed è in carico al team clinico che grazie agli strumenti forniti è in grado di creare ripetizioni dedicate a specifici pazienti, nonché riutilizzare stimoli precedentemente configurati in diverse attività e ripetizioni.

Sia gli stimoli che le ripetizioni sono associabili ad un oggetto chiamato "tema" ovvero un tag volto a raccogliere contenuti che abbiano un contenuto simile per facilitare la costruzione di ripetizioni ed esercitazioni specifiche. Tali oggetti tema sono anch'essi completamente configurabili tramite CMS.

2 DB

Il Database, di seguito rappresentato da uno schema ER nell' allegato "Allegato 1 – ER.pdf", vede nell'entità user e nell'entità product (corrispondente ad una attività neuro cognitiva) i suoi nodi principali.

A livello di Database sono rappresentate in modo fedele tutte le entità utilizzate da API, CMS e APP che quindi si traducono in modo fedele in modelli o Classi utilizzate nelle applicazioni per istanziare gli oggetti da esse utilizzati. Fanno eccezione solamente le tabelle di relazione molti a molti create solo per mantenere le relazioni tra entità.

L'entità Audit è l'unica non legata ad altre entità da vincoli relazionali in quanto pure log delle azioni effettuate.

Di seguito una breve spiegazione delle tabelle, si rimanda all'allegato "Allegato 2 -MYSQL Dump struttura e dati.sql" per una visione di dettagli dei campi e delle loro relazioni nonché una versione del Database pulito ma già configurato per le attività già sviluppate.

2.1 Tabelle relative alle utenze

company
id, name, createdAt, updatedAt

Nella tabella *company* vengono definiti i team di terapisti così da isolare i teams in un'ottica multi-tenancy

license
id, name, createdAt, updatedAt

La tabella *license* definisce un supergruppo di utenti trasversale ai team, che può essere utilizzato per definire aspetti grafici diversi o accessi diversi alle attività presenti sulla piattaforma

user
id, username, password, type, clinicianId, companyId, isCompanyAdmin, licenselId, licenseDeadline, name, surname, photo, startupSync, lastSync, appVersion, createdAt, updatedAt, deletedAt

Sebbene la creazione degli utenti terapisti sia a disposizione nel CMS per gli utenti amministratori e la creazione di utenze pazienti sia a disposizione di tutti i terapisti, almeno un utente di tipologia amministratore deve essere configurato a sistema così che possa accedere al CMS e configurare le altre utenze terapisti.

I campi obbligatori per la creazione dell'utente amministratore di company sono *username*, *password*, *type*, *companyId*, *isCompanyAdmin*, *LicenselId*, *LicenseDeadline*, *name*, *surname*. Tramite il campo *licenseDeadline* è possibile configurare un limite ultimo di accesso alla piattaforma.

Nel caso di creazione di utenze di terapisti queste ereditano dall'utenza amministrativa i campi *companyId*, *licenselId* e *licenseDeadline*.

Nel caso di creazione di utenze pazienti queste ereditano dall'utenza terapeuta i campi *companyId*, *licenselId* e *licenseDeadline*, mentre il campo *clinicianId* verrà impostato con l'*id* del terapeuta corrente legando quindi il paziente allo stesso in modo che questo non possa essere gestito da altri terapisti.

registry
id, userId, birthDate, diag, sex, education, note, createdAt, updatedAt

In questa tabella vengono registrati i dati dei pazienti utili alle analisi a posteriori

diagnosis
id, name, description, createdAt, updatedAt

Questa tabella mantiene un elenco delle diagnosi disponibili per l'associazione ad utenze pazienti

user_diagnosis
id, userId, diagnosisId, createdAt, updatedAt

Tabella di relazione che lega pazienti a diagnosi

Domiciliation
id, userId, startDate, endDate, extra, createdAt, updatedAt
host
id, companyId, UUID

Domiciliation e *host* sono due tabelle al momento non utilizzate dal sistema ma che sono state introdotte per funzioni che sono previste nel momento in cui INFORMA passerà ad essere effettivamente multi-tenancy e saranno volte a monitorare l'utilizzo della piattaforma ed evitare utilizzi impropri delle credenziali di accesso.

2.2 Tabelle relative alle attività

product
id, typeId, name, fileName, thumbnail, filePath, version, collection, isMotionBased, description, deepening, createdAt, updatedAt

In questa tabella si elencano le attività offerte dalla piattaforma

setting
id, productId, key, name, index, type, father, defaultValue, choices, conditional, inputs, isFiltered, isRequired, minLength, maxLength, minValue, maxValue, valueStep, customCheck, customCheckMessage, visible, isStimulus, stimulusType, createdAt, updatedAt

Questa tabella risulta fondamentale nell'integrazione delle attività nell'APP e CMS.

Si elencano qui infatti i parametri di configurazione che andranno a creare dinamicamente le interfacce di configurazione nel CMS e gli oggetti di input per APP.

È possibile prevedere un livello di annidamento di tali settaggi specificando l'id di un altro setting nella colonna *father*.

Di interesse particolare sono le colonne

choices, dove è possibile quando si definisce il type "*choice*" specificare un elenco di scelte per il settaggio come array di oggetti con valore "*key*" e label "*name*"

conditional, dove è possibile specificare una condizione relativa agli altri setting del product che determina la visibilità o meno del setting corrente

customCheck che permette di inserire una espressione da validare all'interno dell'attività, la quale se non passata blocca il salvataggio della ripetizione da parte del terapeuta con messaggio definito nella colonna *customCheckMessage*

scale
id, productId, key, name, min, max, step, labels, weight, inputUI, sortingUI, isFree, createdAt, updatedAt

In questa tabella sono definiti i settaggi di configurazione a livello di attività, ovvero quei settaggi che saranno uguali in tutte le ripetizioni e sono impostati di volta in volta al momento della pianificazione dell'attività

level
id, productId, name, settings, createdAt, updatedAt

Ogni attività deve avere un livello di default che definisca un insieme di scale come oggetto nella colonna settings

gap
id, productId, key, name, createdAt, updatedAt

Questa tabella funge come repository di label per i parametri di performance prodotti dalle attività

setup
id, userId, productId, levelName, levelSettings, repetitionSettings, createdAt, updatedAt

In questa tabella viene salvato l'ultimo stato di configurazione utilizzato da un paziente in un'attività

2.3 Tabelle delle entità della libreria

theme
id, name, thumbnail, createdAt, updatedAt

Semplice elenco dei temi creati e delle immagini ad essi assegnate

stimulus
id, name, type, content, createdAt, updatedAt

Gli *stimuli* creati dai terapeuti.

type può essere text, audio, video, image.

mentre *content* mantiene il testo in caso di stimolo testuale o il nome del file uploadato nel caso di contenuto media.

stimulus_theme
id, themeId, stimulusId

tabella di relazione tra *stimulus* e *theme*

repetition
id, name, settings, stimuli, productId, createdAt, updatedAt

In questa tabella vengono mantenuti i *setting* relativi ad una ripetizione come oggetto JSON all'interno del campo *settings* e un elenco dei nomi dei file media utilizzati come array JSON nel campo *stimuli*, al fine di facilitare l'opera di download da parte di APP nel momento in cui scarica una nuova pianificazione.

La sedizione di mantenere i settings come campo JSON delega al lato applicativo la gestione di tale struttura dati e la sua interpretazione, rendendo quindi più facile a livello di database gestire attività in modo indipendente dalla loro reale implementazione al fine di rendere la piattaforma estendibile con altre esercitazioni senza necessità di intervenire sulla struttura del database.

repetition_theme
id, themeId, repetitionId

Tabella di relazione tra *repetition* e *theme*

2.4 Altre tabelle

navigation

id, section, name, type, parentId, licenseld, companyld, userld, productld, isActive, createdAt, updatedAt
--

In questa tabella vengono elencati i contenuti della piattaforma (attività) e nel caso si voglia possono essere limitati per company (*companyld*), license, (*licenseld*) o user (*userld*), o eventualmente semplicemente disabilitati tramite la colonna *isActive*.

audit

id, userld, action, targetResource, targetData, status, createdAt

In questa tabella vengono riportate le azioni di accesso ai dati da parte degli utenti, la tipologia di dati e i dati stessi che sono stati richiamati, creati, aggiornati o eliminati

message

id, userld, text, isChecked, link, isLocked, createdAt, updatedAt

Tabella di archiviazione dei messaggi delle utenze terapisti

diary

id, userld, text, date, createdAt, updatedAt
--

Tabella di archiviazione dei messaggi delle utenze pazienti

session

id, userld, startedAt, endedAt, createdAt, updatedAt
--

In questa tabella vengono archiviate le sessioni di login degli utenti.

activity

id, userld, productld, themeld, scheduledFrom, scheduledTo, levelName, levelSettings, value, detail, startedAt, endedAt, isDelayed, isCompleted, targetVersion, complexity, performance, balance, isLocked, createdAt, updatedAt, deletedAt

In questa tabella viene mantenuta la pianificazione delle esercitazioni per ogni utente. I risultati di performance e di impostazione globale dell'attività vengono mantenute come file JSON in *value*, *detail* e *levelSettings*. Così come già in *repetition* la decisione di mantenere tali informazioni in formato JSON nasce dalla necessità di creare una base dati indipendente dall'implementazione delle attività stesse.

Le ripetizioni collegate alle attività sono mantenute come relazioni allo stato attuale ma nello sviluppo futuro vedranno la loro cristallizzazione in un campo JSON in questa tabella, dando quindi la possibilità ai terapisti di modificare o eliminare tali ripetizioni dopo aver assegnato una esercitazione senza per questo andare a compromettere la stessa.

activity_repetition

id, activityld, repetitionld

Questa tabella mantiene le relazioni tra le attività e le ripetizioni ad esse collegate.

3 API

Il server Api è sviluppato in NodeJs e fornisce l'accesso alle proprie API tramite routes predefinite e attinge al Data Layer tramite un pattern a repository.

Le funzioni più complesse, di maggiore importanza e/o che sono utilizzate in più routes, vedono lo sviluppo di un modulo dedicato.

Sono presenti dei middleware di sicurezza, alcuni forniti da plugin, mentre altri, più semplici, implementati direttamente.

Tutte le comunicazioni con le API avvengono tramite chiamate HTTPS e l'accesso alle API è mediato da un protocollo di autorizzazione basato su JSON Web Tokens (JWT) rilasciati al momento di login.

Sono escluse da questo meccanismo di autorizzazione le chiamate all'Api check (per il controllo della connessione) e quelle di login.

3.1 Server e App

Il file server.ts lancia il server che effettua il bind delle chiamate web e configura l'applicazione principale come gestore di tale chiamate.

Questo file contiene la configurazione di base del server creato da API, tra cui la gestione della porta e delle chiavi SSL per il funzionamento del protocollo HTTPS.

La classe App contiene l'effettiva applicazione che gestisce le chiamate all'API, qui troviamo la registrazione di routes e middlewares.

In environment di sviluppo (dev) l'applicazione fornisce anche accesso alla path statica di "upload" che normalmente è caricata sotto il web server che fornisce accesso a CMS.

3.2 Middlewares:

I Middleware di principale rilevanza sono helmet ed expressJWT.

Helmet è un plugin per server NodeJS che implementa automaticamente molteplici Header HTTP che migliorano la sicurezza di API.

ExpressJWT è un plugin che gestisce le autorizzazioni di accesso alle route, basandosi su protocollo JWT (Json Web Tokens), inizialmente staccati dal modulo "Auth".

Tra i middleware usati citiamo anche uno dedicato all'implementazione di CORS custom, alcuni dedicati alla gestione dei limiti massimi di upload nonché disabilitazione della cache.

3.3 Routes

Le classi di route implementano l'elaborazione prevista per le chiamate HTTPS, tramite metodi dedicati di post, get, put e delete che implementano l'API REST.

I metodi post prevedono in ingresso oggetti specificati da classi modello associate, le stesse classi che sono usate dai metodi get per generare gli oggetti che compongono l'output.

I metodi di post, put e delete restituiscono un oggetto custom che riporta un messaggio di successo e ove necessario l'entità coinvolta nell'azione.

Tutte le route hanno funzionalità minima al loro interno, demandano la logica delle operazioni a moduli o al substrato di repositories.

Le routes di *check*, *download*, *upload*, autenticazione e sincronizzazione non rispettano questi ultimi punti.

check è una route di servizio utilizzata dai client per verificare la connessione ad internet.

La route di *download* recupera la posizione fisica dello zip dell'attività dalla configurazione mantenuta a DB e inizia lo stream di download.

La route di *upload* utilizza il plugin formidable per ricevere il file e salvarlo nella cartella configurata nel file di environment.

La route di autenticazione riceve una chiamata post e utilizza il modulo *auth* per verificare la correttezza dell'autenticazione e staccare un token di autorizzazione per inviarlo al client o eventualmente inviare un errore di autenticazione.

Le route di sincronizzazione per terapeuta e paziente ricevono in ingresso una collezione di dati di utilizzo e configurazione e producono in uscita i dati raccolti dagli altri utenti e di configurazione attuale utilizzando i rispettivi moduli di sincronizzazione.

Queste route rappresentano il principale punto di accesso ai dati per APP mentre CMS riceve da *sync* i dati dei pazienti e di configurazione ed ottiene i dati della libreria e pianificazione tramite chiamate sincrone alle route di *theme*, *stimulus*, *repetition* e *activity*.

3.3.1 Elenco completo delle routes api

check

percorso: '/check', metodo: 'get'

auth

percorso: '/auth/patient', metodo: 'post'

percorso: '/auth/clinician', metodo: 'post'

sync

percorso: '/sync/patient', metodo: 'post'

percorso: '/sync/clinician', metodo: 'post'

download

percorso: '/download', metodo: 'post'

upload

percorso: '/upload', metodo: 'post'

patient

percorso: '/patient', metodo: 'post'

percorso: '/patient/:id', metodo: 'put'

percorso: '/patient/:id', metodo: 'delete'

clinician

percorso: '/clinician', metodo: 'get'

percorso: '/clinician', metodo: 'post'

percorso: '/clinician/:id', metodo: 'put'

percorso: '/clinician/:id', metodo: 'delete'

diagnosis

percorso: '/diagnosis', metodo: 'get'

message

percorso: '/messagè', metodo: 'post'

percorso: '/message/:id', metodo: 'put'

percorso: '/message/:id', metodo: 'delete'

theme

percorso: '/themè', metodo: 'get'

percorso: '/themè', metodo: 'post'

percorso: '/theme/:id', metodo: 'put'

percorso: '/theme/:id', metodo: 'delete'

stimulus

percorso: '/stimulus', metodo: 'get'
percorso: '/stimulus', metodo: 'post'
percorso: '/stimulus/:id', metodo: 'put'
percorso: '/stimulus/:id', metodo: 'delete'

repetition

percorso: '/repetition', metodo: 'get'
percorso: '/repetition/:id', metodo: 'get'
percorso: '/repetition', metodo: 'post'
percorso: '/repetition/:id', metodo: 'put'
percorso: '/repetition/:id', metodo: 'delete'

activity

percorso: '/activity', metodo: 'post'
percorso: '/activity/:id', metodo: 'put'
percorso: '/activity/:id', metodo: 'delete'
percorso: '/activity/:id/clonè', metodo: 'post'
percorso: '/activity/download', metodo: 'post'

3.4 Moduli

Tra i moduli troviamo tre moduli di servizio utilizzati in più punti: *DB* e *ErrorResponse* e *Auditor*.

Tali moduli esportano metodi utili e compatti per l'utilizzo nell'applicazione al fine di evitare ridondanza di codice e migliorare la lettura delle classi quali le repository.

Di particolare importanza sono i moduli *sync-clinician* e *sync-patient* che elaborano i dati in ingresso dai client e recuperano le nuove informazioni presenti a DB per l'invio ai client stessi.

Il modulo *Auth* fornisce la logica di autenticazione del paziente tramite controllo delle credenziali confrontandole con quanto salvato a DB.

Nel caso sia abilitata la configurazione pertinente, questo modulo verifica anche che il collegamento dell'utente sia avvenuto da una macchina registrata tramite licenza e salvata nelle configurazioni a DB. Questa funzionalità nasce al fine di evitare l'utilizzo improprio di credenziali di accesso nell'utilizzo futuro del sistema INFORMA in modalità multitenancy.

Completa l'elenco dei moduli la classe *Report* il cui compito è la generazione del file di report delle attività per l'esportazione dei dati nel CMS.

3.5 Preparazione eseguibili

Nella cartella principale dei sorgenti è presente una cartella "Commands" che contiene file script *bash* per facilitare il testing e l'esportazione dell'applicazione per l'utilizzo in ambiente di produzione.

Tali script sono legati all'applicazione tramite apposita configurazione nel file *packages.json*. Ad esempio è possibile testare localmente l'applicazione lanciando "npm run serve" o esportare una nuova versione di produzione con "npm run build prod X.Y.Z" dove X.Y.Z è la versione semantica che si vuole impostare nei file creati.

Tramite il comando di build l'applicazione sviluppata in Typescript viene transpilata in JS secondo quanto configurato in *tsconfig.json* e i files generati sono posizionati sotto la cartella *dist*.

Come poi anche per i client sono presenti 3 environments di esportazione: dev, stag e prod, configurabili per un ambiente di sviluppo, uno di test e uno di produzione rispettivamente.

I files di configurazione degli environments contengono

- le informazioni di collegamento al Database
- La posizione delle cartelle di upload temporaneo e definitivo
- La posizione della cartella contenente i files zip delle attività configurate a DB
- Il secret utilizzato per la creazione degli auth token

4 Clients

I client INFORMA sono sviluppati con il framework Angular ed esportati per la piattaforma scelta.

I client, nonostante la diversa destinazione di esportazione, condividono un modulo contenente i modelli e le funzionalità in comune, tale modulo (da qui in avanti riferito come “UNIVERSAL”) è mantenuto indipendentemente dai client ma il suo sviluppo è ovviamente strettamente legato all’evoluzione delle funzionalità comuni dei client.

Le applicazioni prevedono la possibilità di caricamento di stili css personalizzati legati alla *license* configurata a Database così da agevolare il passaggio a multi-tenancy della piattaforma INFORMA.

Le applicazioni Angular vedono l’accesso alle loro varie sezioni tramite route locali Angular, definite nella classe *AppRoutingModule*, dove le route sono legate ai Componenti pagina che forniscono le funzionalità collegate.

Per scelta applicativa, i popup delle applicazioni sono gestite tramite un servizio collegato che estende un servizio comune *VisibilityService* il quale mantiene lo stato attuale della property *isVisible* utilizzata poi dai componenti dei popup per gestire la propria comparsa. Entrambi i client vedono la presenza di una sezione dedicata alla loro esportazione, sotto la cartella “builders”. Una discussione dedicata alle funzionalità disponibili è rimandata ai capitoli relativi all’esportazione eseguibili per ognuno dei due client.

4.1 CMS

CMS è un’applicazione web Angular fornita tramite web Server per fruizione tramite browser.

Le principali funzionalità offerte includono:

- Gestione utenti clinici (riservato per utenti amministratori di team)
- Gestione pazienti
- Gestione libreria (temi, stimoli e ripetizioni)
- Gestione programmazione esercitazioni
- Gestione delle comunicazioni con il paziente
- Visualizzazione dati di utilizzo e performance
- Esportazione dei dati di allenamento

Questa applicazione vede la maggior parte dei dati sincronizzati all’avvio o tramite apposita funzione di sincronizzazione. I dati relativi alla libreria vengono invece recuperati in modalità sincrona all’accesso della pagina relativa.

4.1.1 Moduli principali

4.1.1.1 Core

In questo modulo vengono riportate le funzionalità di base dell’applicazione CMS. *MainComponent* contiene il componente di routing (*router-outlet*) e pertanto ha la funzione di wrapper per le varie aree e di organizzazione del layout, al momento organizzato semplicemente in barra del menu superiore e navigazione nella restante parte dello schermo.

Il componente *NavComponent* gestisce la barra superiore, e fornisce le funzionalità grazie al supporto dei componenti *UserMenuComponent* per logout, gestione dati personali e

accesso admin alla gestione delle utenze cliniche (mediato da *AdminGuard*), *PatientButtonComponent* e *PatientMenuComponent* per accesso alla gestione delle anagrafiche pazienti, la selezione del paziente per l'impostazione della terapia e l'export massivo di dati di molteplici pazienti.

I componenti *DashboardPageComponent*, *DeskPageComponent* e *LibraryPageComponent* forniscono semplicemente una navigazione statica all'interno dell'applicazione rispettivamente nelle pagine iniziale, di gestione della terapia paziente e gestione della libreria.

Il componente *DeskPageComponent* è accessibile solo previa scelta di un paziente nel menu superiore, tale accesso è modulato da *PatientGuard*, sotto il modulo *Patient*

4.1.1.2 Communication

Questo modulo gestisce la visualizzazione e composizione di messaggi da parte dei clinici (*message*) e pazienti (*diary*).

L'accesso a tale funzionalità è mediata da *PatientGuard*, sotto il modulo *Patient*

4.1.1.3 Calendar

Al modulo *Calendar* è demandata la gestione della programmazione delle attività nel tempo da parte del terapeuta e il relativo salvataggio online.

Queste funzionalità sono implementate tramite un wrapper settimanale (*CalendarPageComponent*) di un componente che mostra e gestisce le impostazioni per un singolo giorno (*CalendarDayComponent*).

Tramite il popup dedicato all'inserimento di una nuova pianificazione è possibile accedere ai componenti di gestione di temi e ripetizioni (e quindi di stimoli), in questo modo il componente *calendar* può essere il punto di accesso alla creazione di elementi della libreria.

4.1.1.4 Interchange

Questo modulo si occupa di ricevere i dati di input dalla sincronizzazione e preparare i dati di output da inviare nella sincronizzazione. Effettua queste operazioni appoggiandosi a metodi specifici offerti dai servizi associati alle varie entità.

4.1.1.5 Library

In questo modulo troviamo i componenti di visualizzazione e modifica dei tre elementi principali della libreria: *Theme*, *Stimulus* e *Repetition*.

In questo modulo riportiamo anche se non esistente in produzione un componente di manutenzione, *ProductsListComponent*, che permette di aggiornare la versione degli zip delle attività caricate senza intervenire a DB.

4.1.1.6 Patient

Nel modulo *Patient* sono contenute le funzionalità di gestione delle anagrafiche dei pazienti nonché il mantenimento dello stato attuale della scelta corrente di paziente per l'accesso alla gestione terapia, comunicazioni e visualizzazione dati di utilizzo (modulo *performance*), il cui accesso è modulato da *PatientGuard*

4.1.1.7 Performance

Il modulo *performance* raccoglie le funzionalità di esplorazione delle attività del paziente corrente, con relative funzionalità di export dedicato nel componente *ActivityPageComponent* e l'accesso al componente di export massivo (*ExportPageComponent*).

Sia l'esportazione paziente che l'esportazione massiva fanno affidamento ad un Api dedicata

per la generazione del file di esportazione disponibile nella route “/activity/download” di API.

4.1.1.8 *Universal*

Questo modulo è gestito in un progetto separato e incluso in entrambi CMS ed APP (Vedi Capitolo UNIVERSAL)

4.1.2 Preparazione eseguibili

Il client INFORMA dedicato al team clinico (CMS) è un client web, la cui esportazione dipende anche dall’environment di esportazione, scelto tra i tre disponibili (dev, stag e prod). Nel CMS, gli environments riportano solo gli URL di collegamento per le api remote e la chiave di crittografia per lo storage locale.

Come anticipato nel capitolo generale dei clients, le funzionalità di preparazione degli eseguibili sono disponibili nella cartella builders, dove nella sotto cartella commands troviamo eseguibili bash legati all’applicazione tramite il file packages.json.

Per testare l’applicazione si possono utilizzare i comandi npm run dev, npm run dev-stag, npm run dev-prod per testare il CMS collegandolo ad API in ambienti rispettivamente di dev, stag e prod.

Per esportare una versione sono disponibili i comandi npm run build-stag e npm run build per esportare rispettivamente le versioni di staging (test) e produzione. I comandi di build accettano anche un parametro opzionale di versione in formato numerico secondo il paradigma di versionamento semantico MAJOR.MINOR.PATCH.

L’azione di build transpila il codice sorgente Typescript in codice Javascript secondo quanto configurato in app/tsconfig.json e app/angular.json e posiziona il codice finalizzato in builders/www

4.2 APP

L'applicazione dedicata ai pazienti è un client sviluppato grazie ad Angular Framework ed esportato in ambienti desktop e mobile grazie a strumenti dedicati che fungono da wrapper per l'ambiente nativo.

Questo approccio permette di mantenere un solo codice sorgente con minimi adattamenti necessari per il funzionamento sulle varie piattaforme (questi aspetti verranno approfonditi nella descrizione del modulo *platform* di UNIVERSAL e nel capitolo dedicato alla preparazione degli eseguibili di APP)

4.2.1 Comunicazione con API

APP è stata studiata per poter operare senza connessione di rete una volta effettuato il login e per scambiare informazioni con API non appena una connessione sia disponibile.

Per ottimizzare lo scambio di informazioni nel minor tempo possibile, anche in condizioni in cui la connessione disponibile sia degradata, la comunicazione con API è stata relegata ad un singolo punto di accesso, fatta eccezione per il login.

Il singolo punto di comunicazione per APP è servito da *SyncService*, all'interno del modulo *sync* di UNIVERSAL.

Questo servizio fa affidamento su *InterchangeService*, del modulo *Core* per la preparazione dei dati da inviare ad API e per il salvataggio di quanto restituito.

Il collegamento avviene sulla route API `"/sync/patient"` con metodo POST.

Il cliente APP invia in formato JSON un oggetto custom *inputData* contenente l'utente corrente, i messaggi ricevuti ed inviati, le sessioni di accesso, le ultime impostazioni utilizzate per ogni esercizio e l'elenco delle esercitazioni pianificate aggiornato con i dati di utilizzo/performance.

L'oggetto inviato ha la seguente struttura:

```
{
  "user": {...},
  "sessions": [...],
  "setups": [...],
  "activities": [...],
  "messages": [...],
  "diaries": [...]
}
```

L'oggetto inviato è costituito da un oggetto *user*, e cinque collezioni di oggetti *sessions*, *setups*, *activities*, *messages* e *diaries*.

La struttura di questi oggetti è definita nei file di modello presenti nel codice sorgente di CORE e APP.

Diary è l'unico modello di questi che risiede all'interno di APP, mentre i restanti risiedono all'interno del modulo condiviso CORE.

I modelli sono disponibili nel codice sorgente ai seguenti percorsi:

CORE

user\user.ts

session\session.ts

user\setup.ts

activity\activity.ts

message\message.ts

APP

src\app\communication\diary.ts

API restituisce un oggetto JSON contenente l'oggetto utente, i messaggi inviati e ricevuti, l'elenco delle esercitazioni pianificate aggiornato con le nuove impostazioni dai terapisti, l'elenco dei temi, le ultime impostazioni utilizzate per ogni esercizio (per gestire casi in cui lo stesso utente possa aver usato un dispositivo diverso), l'elenco dei contenuti a lui consentiti secondo impostazioni di *company* o *license*, l'elenco delle attività a lui assegnate corredate dell'elenco di *settings*, *levels*, *scales* e *gaps* corrispondenti

L'oggetto restituito ha la seguente struttura:

```
{
  "user": {},
  "navigations": [],
  "products": [{...
    "levels": [...],
    "scales": [...],
    "settings": [...],
    "gaps": [...]
  }, ...
],
  "activities": [],
  "setups": [...],
  "themes": [...],
  "messages": [],
  "diaries": []
}
```

Essendo il modello dell'oggetto restituito da API un superset di quanto inviato da APP, differenziato quindi solo in contenuto nel momento in cui dall'ultima sincronizzazione siano stati inseriti nuovi dati da terapeuta o da paziente, per entrambe le comunicazioni valgono le seguenti descrizioni degli oggetti.

User è un oggetto JSON che riporta le informazioni utente mappate 1:1 con la tabella *user* a database, con le seguenti eccezioni

- il campo *CompanyId* è sostituito con un oggetto *company* che riporta le informazioni del team o azienda corrente come inserite nella tabella *company* a database
- aggiunta del campo *registry*, mappato ad un oggetto che contiene le informazioni corollarie sull'utente, ovvero quanto contenuto nella tabella *registry* a database filtrato per l'utente corrente
- aggiunta del campo *domiciliation*, mappato ad un oggetto contenente quanto riportato nella tabella omonima a database. Questo oggetto è al momento inutilizzato nelle funzionalità della piattaforma ed è stato creato come anticipo della funzionalità a backlog per la gestione delle utenze disattivate. Per garantire la funzionalità attuale per ogni utente viene creata insieme all'utenza una *domiciliation* con scadenza al 31/12/2099.

navigation, *activities*, *setups*, *themes*, *messages* e *diaries* sono collezioni di oggetti JSON che mappano fedelmente le omonime tabelle a database facendo riferimento agli omonimi modelli presenti nel progetto sotto la cartella API/models, mentre *products* contiene al suo interno 4 oggetti che vedono la loro mappatura dalle omonime tabelle a database filtrate per il prodotto selezionato.

navigations è una collezione di oggetti di servizio per abilitare o disabilitare l'accesso a specifiche esercitazioni o lo strumento di comunicazione.

setups è una collezione di oggetti di servizio per memorizzare le ultime impostazioni usate da un paziente su una determinata esercitazione.

themes è un oggetto di servizio contenente una collezione dei temi esistenti per mappare nomi e thumbnails agli id dei temi riportati dalle attività

products è un oggetto di servizio contenente l'elenco delle esercitazioni esistenti, con i loro livelli predefiniti (*levels*), i modelli delle impostazioni di complessità globale (*scales*), i modelli delle impostazioni di complessità specifica per ripetizione (*settings*) e i modelli dei parametri di performance misurati (*gaps*)

messages e *diaries* sono collezioni di oggetti che contengono rispettivamente i messaggi dei terapeuti e quelli degli utenti per le comunicazioni.

activities è una collezione di oggetti che riporta sia le esercitazioni future impostate dai terapeuti, sia le esercitazioni svolte con i risultati di performance.

Activities, *messages*, e *diaries* riportano solo dati relativi all'utente corrente.

L'oggetto di risposta API viene costruito grazie ai modelli presenti nel codice sorgente API sotto il percorso *src\models*.

Un esempio di risposta API può essere esplorato nell'Allegato A.

Come già specificato nella descrizione di API, la comunicazione avviene tramite canale HTTPS, con autenticazione gestita tramite JSON Web Tokens (JWT) rilasciati in fase di login. A differenza del token delle utenze terapisti, il token per l'applicazione paziente è rilasciato senza una data di scadenza, dato che il possesso dell'apparecchio mobile e le logiche di accesso allo stesso garantiscono un certo grado di affidabilità dell'autenticazione di accesso alla piattaforma.

4.2.2 Moduli principali

4.2.2.1 Core

In questo modulo vengono riportate le funzionalità di base dell'applicazione APP.

MainComponent contiene il renderizzatore del componente di routing (router-outlet) e pertanto ha la funzione di wrapper per le varie aree e di organizzazione del layout, al momento organizzato semplicemente in barra del menu superiore e navigazione nella restante parte dello schermo.

Il componente *NavComponent* gestisce la barra superiore, e fornisce le funzionalità grazie al supporto dei componenti *UserMenuComponent* per logout.

Il componente *DashBoardComponent* rappresenta la landing page per l'utente paziente e si occupa di effettuare la sincronizzazione, scaricare contenuti di libreria collegati alle esercitazioni programmate ed eventualmente scaricare gli esercizi se mancanti (primo avvio) o aggiornati.

DashBoardComponent dopo aver verificato la presenza di tutti i file necessari, raccoglie le attività programmate per il giorno corrente (o precedenti) e le raggruppa per tema, al fine di

presentare al paziente la scelta del tema da affrontare.

Tutte le operazioni sopraelencate vengono svolte grazie al supporto di servizi dedicati presenti nel modulo UNIVERSAL.

In questo modulo risiede anche il servizio che ha il compito di preparare le informazioni da inviare ad API e gestire il salvataggio in locale di quanto restituito: *InterchangeService*.

Questo servizio viene richiamato dal modulo *sync* di UNIVERSAL.

4.2.2.2 *Communication*

Questo modulo gestisce la visualizzazione e composizione di messaggi da parte dei clinici (*message*) e pazienti (*diary*).

4.2.2.3 *Training*

Il modulo training presenta l'elenco delle esercitazioni disponibili sotto un determinato tema scelto nel *DashBoardComponent*

4.2.2.4 *Universal*

Questo modulo è gestito in un progetto separato e incluso in entrambi CMS ed APP (Vedi Capitolo UNIVERSAL)

4.2.3 Preparazione eseguibili

Il client INFORMA dedicato ai pazienti (APP) è un client sviluppato come applicazione web ma esportato per piattaforme native grazie al supporto di tool dedicati che fungono da wrapper per le piattaforme target.

Nel caso di piattaforme Desktop, si è scelto di fare affidamento su Electron, mentre per le piattaforme mobile il tool di riferimento è Cordova.

L'esportazione di APP fa affidamento a dei files che configurano l'environment di esportazione, sono disponibili, come nelle altre applicazioni, dev, stag e prod.

In APP, gli environments definiscono gli URL di collegamento per le api remote e la chiave di crittografia per lo storage locale.

Come anticipato nel capitolo generale dei clients, le funzionalità di preparazione degli eseguibili sono disponibili nella cartella builders, dove nella sotto cartella commands troviamo eseguibili bash legati all'applicazione tramite il file packages.json.

Per testare l'applicazione si possono utilizzare i comandi `npm run dev`, `npm run dev-stag`, `npm run dev-prod` per testare APP come applicazione Web collegandola ad API in ambienti rispettivamente di development, staging e production. Questa modalità permette di accedere velocemente all'applicazione ed effettuare debug direttamente collegato ai codici sorgenti grazie al sourcemapping offerto da webpack. In questa modalità, tuttavia, non è possibile al momento utilizzare le attività in quanto lo storage web non supporta tale funzionalità.

Per testare in l'applicazione finale è possibile usare il comando "go" per lanciare il locale o su un dispositivo collegato via USB. In questo caso dobbiamo specificare la piattaforma target e l'environment che vogliamo utilizzare, pertanto esempi di lancio sono `npm run go desktop dev` oppure `npm run go android prod`.

Il comando dedicato alla compilazione di una versione installabile si comporta analogamente a al comando precedente con l'aggiunta del parametro opzionale del numero di versione in formato numerico secondo il paradigma di versionamento semantico MAJOR.MINOR.PATCH.

Esempi di esportazione sono quindi

`npm run build desktop stag` oppure `npm run build android prod 1.0.8`

L'azione di build transpila il codice sorgente Typescript in codice Javascript secondo quanto configurato in `app/tsconfig.json` e `app/angular.json` e posiziona il codice finalizzato in `builders/www`, dopodiché il wrapper dedicato alla piattaforma specifica per l'esportazione genera un eseguibile di installazione e lo posiziona all'interno della cartella "build".

4.3 UNIVERSAL

Nel modulo Universal sono comprese le funzionalità comuni alle due applicazioni, tool di servizio comuni ma anche funzionalità al momento utilizzate da una singola applicazione che sono state spostate in questo modulo perché sono attivi sviluppi di enhancement che prevedono il loro utilizzo in entrambe

(E.g. Componente *product/workshop-page* e modulo download che verranno utilizzati per fornire al team clinico un'anteprima del risultato finale delle impostazioni scelte per le ripetizioni o modulo Message per la gestione delle notifiche di messaggi pazienti per il terapeuta o modulo *Navigation* che sarà utilizzato per garantire l'esplorazione libera dei contenuti da parte del terapeuta)

4.3.1 Moduli principali

All'interno del modulo UNIVERSAL alcuni moduli sono utilizzati solo come interfaccia per garantire il funzionamento parallelo in entrambe le applicazioni nonostante poi tali moduli prevedano una gestione dedicata per le applicazioni CMS ed APP, questo è il caso per i moduli *Activity* e *Library*.

Sono raccolti in Universal anche i principali moduli di servizi, ovvero quei moduli che forniscono funzionalità comunemente utilizzate in varie parti dell'applicazione o di puro supporto ad altri, tra questi il modulo *Tool*, *Session* e *Updater*.

4.3.1.1 Tool

Questo modulo raccoglie tutte le funzionalità utilizzate in più punti da parte delle applicazioni ed accessibili tramite componenti dedicati (e.g. *noticeService* e *NoticeComponent*) o funzioni volte a minimizzare il codice utilizzato e migliorare la leggibilità dei moduli che le utilizzano (e.g. *FilterPipe*)

4.3.1.2 Session

Questo modulo è dedicato solamente al tracciamento delle sessioni di utilizzo dell'applicazione al fine di log

4.3.1.3 Updater

Questo modulo, utilizzato al momento solo da APP, verifica l'esistenza di una nuova versione interrogando il file *latest.yml* all'url passato da APP e configurato sotto *environment.deliveryVersionUrl*. Se tale file contiene una versione più aggiornata, provvede ad aggiornare automaticamente l'applicazione se in ambiente desktop grazie alle funzionalità offerte da Electron, mentre mostra un popup all'utente se in ambiente mobile.

4.3.1.4 Visibility

Il servizio *extend/VisibilityService* fornisce il template per i servizi di lancio di tutti i popup dell'applicazione

Seguono i moduli di UNIVERSAL che ricoprono funzioni specifiche e non di servizio

4.3.1.5 Auth

Questo modulo raccoglie le funzionalità di autenticazione e memorizzano in locale tali informazioni per il prossimo avvio di APP.

4.3.1.6 Download

Questo modulo oltre a fornire le funzionalità di download dei files delle attività e delle ripetizioni, mantiene anche le versioni attuali delle attività in modo da rilevare eventuali aggiornamenti delle stesse alla sincronizzazione successiva.

Durante il download delle attività questo modulo utilizza il servizio Zipper (modulo Tools) per decomprimere i files ricevuti in una cartella virtuale dedicata

4.3.1.7 Product

Nel modulo Product risiede una delle funzionalità principali di APP.

Insieme a servizi e modelli collegati alla gestione dell'elenco delle attività infatti, in questo modulo è fornito il componente *WorkshopPageComponent*.

Questo componente dopo aver caricato le configurazioni di lancio per l'attività corrente, comprendenti tutte le ripetizioni, carica i files dell'attività in un iFrame, lo porta a fullscreen ed invia la configurazione tramite l'Api Web *postMessage* per comunicare in modo sicuro tra due finestre che vengono a tutti gli effetti considerate come cross-origin. Dopodichè si mette in attesa tramite l'Api *onMessage*, che riceverà i risultati di performance da parte dell'attività una volta completata.

Questo componente, nel caso l'attività corrente sia configurata a DB come richiedente le informazioni di DeviceMotion, si preoccuperà anche di inoltrare quest'ultime, legandole alla finestra corrente. In questo modo, l'iFrame dell'attività lanciata potrà accedere liberamente a tali informazioni.

4.3.1.8 User

Il modulo User detiene le informazioni legate all'utente correntemente loggato al sistema, ed in particolare anche la licenza di utilizzo. Al momento questa funzionalità non viene utilizzata ma è prevista per l'utilizzo di INFORMA in ambiente di produzione e specialmente in ambito multi-tenancy.

4.3.1.9 Sync

Il modulo di sincronizzazione fornisce metodi compatti ed universali di sincronizzazione che richiamano l'*interchangeService* specifico delle applicazioni CMS e APP.

4.3.1.10 Platform

Il modulo *platform* è nato per fornire funzionalità specifiche su piattaforme diverse.

Al momento di lancio in test o compilazione per esportazione i files della piattaforma corretta vengono spostati nella cartella *current*.

Le funzionalità principali offerte sono legate al device (accesso alla camera, gestione codice univoco UUID del device, notifiche locali, update, etc..) e tra queste quella più importante è lo storage.

La funzionalità di storage è implementata tramite *indexedDB* nella piattaforma Web, in file nella cartella locale nelle applicazioni Desktop (e.g. AppData/Roaming/InformaClientPatient sotto Windows) e lo storage di device per piattaforme mobile.

Tutti i dati sono crittografati nel momento in cui risiedono nello storage, utilizzando come chiave di crittografia quanto specificato in *environment.encryptedPassPhrase*.

4.3.1.11 Navigation

Questo modulo è stato introdotto per garantire una configurabilità delle visibilità dei contenuti (attività) veicolati da INFORMA.

Tramite opportune configurazioni a DB è possibile infatti legare determinati contenuti a un solo team clinico (company) o ad una sola versione del prodotto (*license*).

E stata predisposta anche la possibilità di filtrare i contenuti disponibili anche in base ad un utente specifico sebbene al momento non esista un sistema di configurazione di tale proprietà.

Allo stesso modo è stata predisposta anche la possibilità di esplorare liberamente le attività ammesse da quanto configurato a Database, in vista della possibilità di offrire tale funzione al terapeuta per verificare le ripetizioni create. Questa funzionalità è implementata tramite il componente *NavigationPageComponent*

5 Attività

Le attività supportate dalla piattaforma sono applicazioni web in grado di essere eseguite direttamente in un browser web senza l'utilizzo di plugin aggiuntivi.

I requisiti per il loro lancio sono l'avere una pagina principale chiamata "index.html" in grado di ricevere ed inviare parametri tramite l'Api Web *PostMessage/OnMessage*.

Le attività fornite nel presente sistema sono sviluppate in JavaScript con il supporto del Framework JQuery.

A livello generale le attività fornite partono da un template generale data la similitudine del flow dei vari esercizi e specificano poi le loro funzionalità in files dedicati ai singoli esercizi. Prima del lancio dell'esercitazione, la funzione *entryPoint* (index.js) si incarica di ricevere i dati da APP e fare un mapping degli stessi in un formato utilizzato poi all'interno dell'attività. In ogni attività quindi troveremo una funzione di base (*exerciseEngine*) che gestirà il lancio dell'esercitazione il passaggio tra le varie ripetizioni e la conclusione dell'esercitazione.

Le ripetizioni sono gestite da una funzione generale (*activityEngine*) che avrà il compito di gestire il flow della ripetizione grazie al supporto di quanto specificato in *activityInput* (che oltre a raccogliere i dati forniti tramite APP definisce anche i default per i comportamenti dell'esercitazione) e produrre risultati secondo quanto specificato da *activityOutput*.

Queste ultime 3 funzioni vengono poi specificate per ereditarietà in ognuna delle attività andando quindi a fornire le funzionalità e caratteristiche specifiche dell'attività corrente.

Alla fine dell'esercitazione, è poi compito di *exerciseOutput* (richiamata da *exerciseEngine*) di raccogliere tutti i risultati generati in *activityOutput* e creare un risultato finale *exerciseOutput* poi invierà ad APP.

6 Allegato A

Esempio di oggetto restituito da API verso APP

```
{
  "user": {
    "id": 315,
    "username": "utente.esempio",
    "password": "5f4dcc3b5aa765d61d8327deb882cf99",
    "type": "patient",
    "clinicianId": 254,
    "companyId": 13,
    "isCompanyAdmin": 0,
    "licenseId": 1,
    "licenseDeadline": "2099-12-30T23:00:00.000Z",
    "name": "Nome",
    "surname": "Cognome",
    "photo": null,
    "startupSync": 1,
    "lastSync": "2022-02-07T10:32:02.000Z",
    "appVersion": "1.1.2",
    "createdAt": "2021-06-30T14:29:16.000Z",
    "updatedAt": "2022-02-07T10:32:02.000Z",
    "deletedAt": null,
    "licenseName": "ReHab",
    "registry": {
      "id": 154,
      "userId": 315,
      "birthDate": "2021-06-29T22:00:00.000Z",
      "taxCode": null,
      "diag": null,
      "education": 0,
      "sex": "M",
      "note": null,
      "employment": null,
      "createdAt": "2021-06-30T14:29:16.000Z",
      "updatedAt": "2021-06-30T14:29:16.000Z"
    },
    "domiciliation": {
      "id": 102,
      "userId": 315,
      "startDate": "2020-12-31T23:00:00.000Z",
      "endDate": "2099-12-30T23:00:00.000Z",
      "extra": null,
      "createdAt": "2021-06-30T14:29:16.000Z",
      "updatedAt": "2021-06-30T14:29:16.000Z"
    },
    "company": {
      "id": 13,
      "name": "TestTeam",
      "createdAt": "2021-02-23T09:19:55.000Z",
      "updatedAt": "2021-02-23T09:19:55.000Z"
    },
    "photoData": null
  },
  "messages": [
```

```

        "id": 42,
        "userId": 315,
        "text": "Perché non hai fatto gli esercizi settimana scorsa?",
        "media": null,
        "link": null,
        "isChecked": 1,
        "isLocked": 1,
        "createdAt": "2021-12-31T23:00:00.000Z"
    },
    ...
],
"navigations": [{
    "id": 1232,
    "section": "Desk",
    "name": "Comunicazioni",
    "type": "Tool",
    "parentId": null,
    "productId": 273
}, {
    "id": 1233,
    "section": "Workshop",
    "name": "Trova La Parola",
    "type": "Product",
    "parentId": null,
    "productId": 233
}, {
    "id": 1239,
    "section": "Workshop",
    "name": "Categorizza",
    "type": "Product",
    "parentId": null,
    "productId": 234
}, {
    "id": 1240,
    "section": "Workshop",
    "name": "Riordina il proverbio",
    "type": "Product",
    "parentId": null,
    "productId": 247
}, {
    "id": 1241,
    "section": "Workshop",
    "name": "Cruciverba",
    "type": "Product",
    "parentId": null,
    "productId": 248
}, {
    "id": 1242,
    "section": "Workshop",
    "name": "Trasforma la parola",
    "type": "Product",
    "parentId": null,
    "productId": 249
}, {
    "id": 1243,
    "section": "Workshop",

```

```

        "name": "Trova l'intruso",
        "type": "Product",
        "parentId": null,
        "productId": 250
    }, {
        "id": 1244,
        "section": "Workshop",
        "name": "Mandala",
        "type": "Product",
        "parentId": null,
        "productId": 259
    }, {
        "id": 1245,
        "section": "Workshop",
        "name": "Puzzle",
        "type": "Product",
        "parentId": null,
        "productId": 260
    }, {
        "id": 1246,
        "section": "Workshop",
        "name": "Idee e pensieri",
        "type": "Product",
        "parentId": null,
        "productId": 261
    }, {
        "id": 1247,
        "section": "Workshop",
        "name": "Riordino del denaro",
        "type": "Product",
        "parentId": null,
        "productId": 262
    }, {
        "id": 1248,
        "section": "Workshop",
        "name": "Curiosita'",
        "type": "Product",
        "parentId": null,
        "productId": 263
    }, {
        "id": 1249,
        "section": "Workshop",
        "name": "Quiz",
        "type": "Product",
        "parentId": null,
        "productId": 264
    }, {
        "id": 1250,
        "section": "Workshop",
        "name": "Cerca il dettaglio",
        "type": "Product",
        "parentId": null,
        "productId": 265
    }, {
        "id": 1251,
        "section": "Workshop",

```

```

        "name": "Descrivi l'immagine",
        "type": "Product",
        "parentId": null,
        "productId": 266
    }, {
        "id": 1252,
        "section": "Workshop",
        "name": "CruciPuzzle",
        "type": "Product",
        "parentId": null,
        "productId": 267
    }, {
        "id": 1253,
        "section": "Workshop",
        "name": "Spesa ed acquisti",
        "type": "Product",
        "parentId": null,
        "productId": 268
    }, {
        "id": 1254,
        "section": "Workshop",
        "name": "Stima numerica",
        "type": "Product",
        "parentId": null,
        "productId": 269
    }, {
        "id": 1255,
        "section": "Workshop",
        "name": "Allenamento motorio 1",
        "type": "Product",
        "parentId": null,
        "productId": 270
    }, {
        "id": 1256,
        "section": "Workshop",
        "name": "Allenamento Motorio 2",
        "type": "Product",
        "parentId": null,
        "productId": 271
    }, {
        "id": 1257,
        "section": "Workshop",
        "name": "Le Avventure di",
        "type": "Product",
        "parentId": null,
        "productId": 272
    }
    ],
    "products": [{
        "id": 233,
        "type": "Workshop",
        "name": "Trova La Parola",
        "fileName": "Definition.zip",
        "thumbnail": "Trova_la_parola.png",
        "filePath": "workshop/Definition.zip",
        "version": 200,

```

```

"collection": "INFORMA",
"isMotionBased": 0,
"description": "Vengono presentate una serie di definizioni alle
quali corrispondono delle parole.",
"deepening": "Attività di ricerca lessicale libera, relativa al
tema trattato.",
"createdAt": "2021-03-16T23:41:08.000Z",
"updatedAt": "2022-01-20T05:19:11.000Z",
"levels": [{
  "id": 209,
  "name": "Demo",
  "settings": {
    "tIsRatingRequired": 1,
    "helpActivationTimeout": 0,
    "textInstructionPosition": 2,
    "helpActivationConsecutiveErrorsNumber": 0
  }
}],
"scales": [{
  "id": 1,
  "productId": 233,
  "key": "helpActivationTimeout",
  "name": "Tempo di esitazione prima di aiuto (s)",
  "min": "0",
  "max": "90",
  "step": 1,
  "labels": null,
  "weight": 1,
  "inputUI": "Slider",
  "sortingUI": null,
  "isFree": 0,
  "createdAt": "2021-03-17T08:51:45.000Z",
  "updatedAt": "2021-06-29T01:29:59.000Z"
}, {
  "id": 2,
  "productId": 233,
  "key": "helpActivationConsecutiveErrorsNumber",
  "name": "Numero di tentativi errati prima di aiuto",
  "min": "0",
  "max": "10",
  "step": 1,
  "labels": null,
  "weight": 1,
  "inputUI": "Slider",
  "sortingUI": null,
  "isFree": 0,
  "createdAt": "2021-03-17T09:04:19.000Z",
  "updatedAt": "2021-06-29T01:29:59.000Z"
}, {
  "id": 3,
  "productId": 233,
  "key": "textInstructionPosition",
  "name": "Posizione del testo",
  "min": "1",
  "max": "3",

```

```

        "step": 1,
        "labels": ["Sinistra", "Centro", "Destra"],
        "weight": null,
        "inputUI": "Radio",
        "sortingUI": null,
        "isFree": 0,
        "createdAt": "2021-05-05T16:37:45.000Z",
        "updatedAt": "2021-06-29T01:29:59.000Z"
    }, {
        "id": 4,
        "productId": 233,
        "key": "tIsRatingRequired",
        "name": "Richiedi valutazione al termine dell'esercizio",
        "min": "1",
        "max": "2",
        "step": 1,
        "labels": ["Sì", "No"],
        "weight": null,
        "inputUI": "Radio",
        "sortingUI": null,
        "isFree": 0,
        "createdAt": "2021-06-30T03:58:57.000Z",
        "updatedAt": "2021-06-30T03:58:57.000Z"
    }
],
"settings": [{
    "id": 1,
    "productId": 233,
    "key": "wordsToGuess",
    "name": "Parole da cercare",
    "index": 3,
    "type": "multiple",
    "father": null,
    "defaultValue": null,
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": 1,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": "this.settings.wordsToGuess.every(w =>
w.word.content.length<20)",
    "customCheckMessage": "Le parole scelte devono avere meno
di 20 caratteri",
    "visible": 1,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:53:16.000Z",
    "updatedAt": "2021-09-24T13:58:08.000Z"
}, {
    "id": 2,
    "productId": 233,

```

```

    "key": "word",
    "name": "Parola",
    "index": 4,
    "type": "text",
    "father": 1,
    "defaultValue": null,
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 1,
    "isStimulus": 1,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:54:28.000Z",
    "updatedAt": "2021-06-30T02:00:06.000Z"
  }, {
    "id": 3,
    "productId": 233,
    "key": "definition",
    "name": "Definizione",
    "index": 5,
    "type": "text",
    "father": 1,
    "defaultValue": null,
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": 1,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 1,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:54:28.000Z",
    "updatedAt": "2021-06-30T02:00:06.000Z"
  }, {
    "id": 4,
    "productId": 233,
    "key": "correctLetters",
    "name": null,
    "index": null,
    "type": "number",

```

```

    "father": 1,
    "defaultValue": "0",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 0,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:54:28.000Z",
    "updatedAt": "2021-05-05T03:57:16.000Z"
  }, {
    "id": 5,
    "productId": 233,
    "key": "visited",
    "name": null,
    "index": null,
    "type": "boolean",
    "father": 1,
    "defaultValue": "false",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 0,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:56:42.000Z",
    "updatedAt": "2021-05-05T03:56:42.000Z"
  }, {
    "id": 6,
    "productId": 233,
    "key": "answered",
    "name": null,
    "index": null,
    "type": "boolean",
    "father": 1,
    "defaultValue": "false",
    "choices": null,
    "conditional": null,

```

```

    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 0,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T03:56:42.000Z",
    "updatedAt": "2021-05-05T03:56:42.000Z"
  }, {
    "id": 9,
    "productId": 233,
    "key": "activityType",
    "name": null,
    "index": null,
    "type": "text",
    "father": null,
    "defaultValue": "Definition",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 0,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T09:51:43.000Z",
    "updatedAt": "2021-05-05T11:21:11.000Z"
  }, {
    "id": 10,
    "productId": 233,
    "key": "textInstruction",
    "name": "Consegna Testuale",
    "index": 1,
    "type": "text",
    "father": null,
    "defaultValue": "Provi a trovare la parola seguendo la
definizione",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": 1,

```

```

    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 1,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T09:51:43.000Z",
    "updatedAt": "2021-06-30T02:00:06.000Z"
  }, {
    "id": 11,
    "productId": 233,
    "key": "audioInstruction",
    "name": "Consegna Audio",
    "index": 2,
    "type": "audio",
    "father": null,
    "defaultValue": "",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,
    "valueStep": null,
    "customCheck": null,
    "customCheckMessage": null,
    "visible": 1,
    "isStimulus": 0,
    "stymulusType": null,
    "createdAt": "2021-05-05T09:51:43.000Z",
    "updatedAt": "2021-07-27T12:57:09.000Z"
  }, {
    "id": 278,
    "productId": 233,
    "key": "basePath",
    "name": null,
    "index": null,
    "type": "text",
    "father": null,
    "defaultValue": "../Library/",
    "choices": null,
    "conditional": null,
    "inputs": null,
    "isFiltered": null,
    "isRequired": null,
    "minLength": null,
    "maxLength": null,
    "minValue": null,
    "maxValue": null,

```

```

        "valueStep": null,
        "customCheck": null,
        "customCheckMessage": null,
        "visible": 0,
        "isStimulus": 0,
        "stymulusType": null,
        "createdAt": "2021-07-28T11:52:33.000Z",
        "updatedAt": "2021-07-28T11:52:33.000Z"
    }
],
"laps": [{
    "id": 170,
    "productId": 233,
    "key": "activityDuration",
    "name": "Durata",
    "isVisible": 1,
    "isMapped": 0,
    "mapping": null,
    "children": null
}, {
    "id": 113,
    "productId": 233,
    "key": "errorsNumber",
    "name": "Errori",
    "isVisible": 1,
    "isMapped": 0,
    "mapping": null,
    "children": null
}, {
    "id": 190,
    "productId": 233,
    "key": "helpActivationsNumber",
    "name": "Aiuti utilizzati",
    "isVisible": 1,
    "isMapped": 0,
    "mapping": null,
    "children": null
}, {
    "id": 207,
    "productId": 233,
    "key": "instructionReadNumber",
    "name": "Ripetizioni consegna",
    "isVisible": 1,
    "isMapped": 0,
    "mapping": null,
    "children": null
}, {
    "id": 227,
    "productId": 233,
    "key": "pPatientScore",
    "name": "Gradimento",
    "isVisible": 1,
    "isMapped": 0,
    "mapping": null,
    "children": null
}, {

```

```

        "id": 250,
        "productId": 233,
        "key": "pSkippedWords",
        "name": "Parole saltate",
        "isVisible": 1,
        "isMapped": 0,
        "mapping": null,
        "children": null
    }, {
        "id": 150,
        "productId": 233,
        "key": "restartsNumber",
        "name": "Restart",
        "isVisible": 1,
        "isMapped": 0,
        "mapping": null,
        "children": null
    }, {
        "id": 130,
        "productId": 233,
        "key": "terminationStatus",
        "name": "Completamento",
        "isVisible": 1,
        "isMapped": 1,
        "mapping": "[\"Completata\", \"Saltata\",
\\\"Interrotta\\\"]",
        "children": null
    }
    ],
    },
    "diaries": [{
        "id": 26,
        "userId": 315,
        "text": "tutto bene",
        "media": null,
        "isChecked": 1,
        "date": "2021-10-15T07:56:35.000Z"
    }, {
        "id": 42,
        "userId": 315,
        "text": null,
        "media": "Registrazione_1644219540152.wav",
        "isChecked": 0,
        "date": "2022-01-01T07:38:27.000Z"
    }
    ]
}

```