ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

Scuola di Ingegneria e Architettura

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

PICTOAI: UN'APPLICAZIONE PER IL SUPPORTO ALL'APPRENDIMENTO MEDIANTE COMUNICAZIONE AUMENTATIVA E ALTERNATIVA RINFORZATA DALL'INTELLIGENZA ARTIFICIALE

Elaborato in
Web Semantico

Relatore
Prof.ssa Antonella Carbonaro

Co-relatori
Prof. Gianluca Moro
Dott. Giacomo Frisoni

Presentata da
Konrad Gómułka

Prima Sessione di Laurea
Anno Accademico 2022 – 2023

KEYWORDS

Comunicazione Aumentativa Alternativa

Intelligenza Artificiale

Cross-Platform Development

Full Stack

Verbalizzazione Knowledge Graph

It gets easier.
Every day it gets a little easier.
But you got to do it every day.
That's the hard part.
But it does get easier.
- da Bojack Horseman

Sommario

Il recente progresso nel campo dell'Intelligenza Artificiale ha aperto la strada ad una vasta gamma di possibili applicazioni innovative, il cui limite è dato solamente dalla capacità di trovare la giusta idea e dalla volontà di realizzarla. Il settore educativo potrebbe beneficiare in particolar modo dell'utilizzo di IA, dal momento che ciò aprirebbe la possibilità di offrire contenuti personalizzati e soluzioni adattabili alle esigenze dei singoli utenti. Questa tesi si concentra sull'impiego dell'Intelligenza Artificiale per aiutare persone con difficoltà nella comunicazione, come disturbi dello spettro autistico, che necessitano di maggior supporto, sia per permettere loro di comunicare efficacemente che per migliorare l'apprendimento adattandosi alle specifiche capacità dell'individuo. Per raggiungere tale obiettivo è stato realizzato un applicativo che integra Intelligenza Artificiale e CAA (Comunicazione Aumentativa Alternativa), un'insieme di tecniche che offre un metodo per comunicare a persone con difficoltà, fornendo loro un supporto accessibile ed efficace all'apprendimento e alla comunicazione.

Introduzione

L'IA (Intelligenza Artificiale) ha ottenuto progressi notevoli negli ultimi anni, rivoluzionando numerosi settori e trasformando il modo in cui viviamo e lavoriamo. Questi progressi sono stati guidati da importanti innovazioni in campi come machine learning, deep learning, NLP (Natural Language Processing) e computer vision. La sua predisposizione ad adattarsi alle specifiche esigenze dei singoli utenti la rendono uno strumento innovativo e potenzialmente rivoluzionario anche nel campo dell'educazione, in particolare nel campo della Comunicazione Aumentativa e Alternativa (CAA).

La CAA propone strategie, tecniche e strumenti in aiuto alle persone con difficoltà nella comunicazione verbale, facilitandola anche con supporti digitali, come software utilizzabili da smartphone o tablet. Un ruolo fondamentale in questo tipo specifico di comunicazione è quello dei simboli grafici, o pittogrammi, che accoppiano una rappresentazione grafica a un'etichetta, così da consentire a queste persone di esprimere più semplicemente il concetto desiderato.

Questa tesi punta alla realizzazione di un supporto software che riesca ad integrare le più recenti tecnologie sviluppate nel campo dell'IA, mettendo a disposizione uno strumento accessibile, all'avanguardia e utile come supporto all'apprendimento per persone con difficoltà comunicative. Un punto cruciale nello sviluppo della tesi è stato lo studio, insieme ad alcuni esperti del settore, delle possibili applicazioni dell'IA a supporto dell'apprendimento mediante CAA, alcune delle quali sono state integrate nel progetto. La tesi si concluderà con una valutazione, da parte degli esperti nel settore, dell'utilizzabilità della soluzione proposta, mettendo in luce tutti i suoi punti di forza e le aree in cui vi è margine di miglioramento. Nel Capitolo 1 viene introdotta la Comunicazione Aumentativa Alternativa, alcune applicazioni dell'IA già esistenti nel settore e una panoramica sui modelli di sviluppo cross-platform più rinomati. Nel Capitolo 2 viene descritta la fase di analisi dei requisti di progetto e esplicata la soluzione proposta. Nel Capitolo 3 viene descritto il prodotto realizzato, esponendone il funzionamento e le possibilità offerte da ciascuna componente. Nel Capitolo 4 viene descritto il processo di validazione del progetto, indicando la valutazione finale fornita dagli esperti del settore. Nel Capitolo 5 viene descritto un lavoro aggiuntivo, complementare al progetto esposto in questa

tesi e utile alla sua componente di IA. Infine, nel Capitolo 5.6 viene descritto il risultato finale, descrivendone punti di forza e limitazioni, suggerendo eventuali sviluppi futuri.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Background

1.1 Comunicazione Aumentativa Alternativa

La **CAA (Comunicazione Aumentativa e Alternativa** comprende una vasta gamma di strumenti, strategie e tecniche progettate per migliorare (aumentativa) e/o compensare (alternativa) la comunicazione di individui che hanno difficoltà ad esprimersi verbalmente. Tali condizioni comprendono disturbi dello spettro autistico, ictus, difficoltà nell'apprendimento, sindrome locked-in o sindrome del chiavistello e malattie come demenza, Parkinson o SLA.[1].

L'utilizzo della CAA prevede due modalità di comunicazione, **assistita** e **non assistita** [2]. La prima, non verbale, avviene tramite gesti, espressioni facciali o lingua dei segni, azioni che necessitano di un adeguato controllo motorio e di un interlocutore in grado di interpretare il messaggio. D'altra parte la comunicazione non assistita richiede un supporto esterno, sia esso analogico (lavagne, tabelle, quaderni e immagini) che digitale, tipicamente software creati ad hoc fruibili tramite smartphone e tablet. La scelta di un approccio non assistito richiede uno studio preliminare che prenda in considerazione le esigenze dello specifico individuo, in modo da offrire gli strumenti adeguati per rendere efficace la comunicazione.

Nella CAA i **simboli pittografici** sono ampiamente utilizzati per accoppiare una rappresentazione grafica ad un'etichetta, così da esprimere il concetto desiderato. Tipicamente vengono adottati in casi di analfabetismo, poiché consentono diversi livelli di complessità nella comunicazione. Allo stato dell'arte non esistono standard affermati ma, uno tra i più comuni, è denominato **ARASAAC**. La popolarità di quest'ultimo deriva da una vasta raccolta di simboli (oltre 11.000), tradotti in più di 23 lingue e distribuiti sotto licenza

open source [3].

1.2 Intelligenza Artificiale

La recente esplosione nello sviluppo delle intelligenze artificiali e la loro predisposizione ad adattarsi alle specifiche esigenze dei singoli utenti, le rendono uno strumento potenzialmente rivoluzionario nel campo dell'educazione. Di recente anche applicazioni per l'insegnamento autonomo di linguaggi rinomate come Duolingo, stanno lavorando all'integrazione delle tecnologie più recenti nel processo di insegnamento. Come descritto in [4], analizzando l'interazione tra l'utente e l'applicativo, si possono individuare le aree in cui esso è più o meno abile, dando quindi la possibilità di concentrarsi su quelle più difficoltose e permettendo un insegnamento più coinvolgente, efficiente e interattivo. Tal discorso vale anche per un applicativo designato per persone con bisogni particolari, utilizzando la CAA rinforzata dall'IA si può facilitare il processo di comunicazione per persone che hanno maggiori difficoltà, fornendo dei mezzi per velocizzare l'intero processo comunicativo e adattando l'insegnamento alle necessità dell'utente.

1.2.1 PictoBERT

Come descritto nel paper di presentazione PictoBERT [5], una schermata di un applicativo di AAC deve semplificare la ricerca e la selezione del pittogramma desiderato. Tali schermate devono prevedere una strategia che semplifichi suddetto compito e, considerando l'elevata quantità di pittogrammi, quella più comunemente adottata consiste nella loro suddivisione in pagine e categorie come "persone", "oggetti", "cibi", etc... La necessità di scorrere più pagine e schermate può però, soprattutto nel caso di utenti con disabilità cognitive, portare l'utente a distrarsi più facilmente complicando il processo di ricerca.
Una possibile strategia per ovviare al problema è la previsione dei pittogrammi successivi in base all'input corrente, fornendoli all'utente in base alla probabilità che siano quelli desiderati. Un approccio simile porta numerosi benefici:

- Riduce il numero di selezioni necessarie per costruire una frase
- Fornisce supporto ortografico per coloro che hanno difficoltà a scrivere correttamente le parole
- Fornisce supporto grammaticale per coloro che hanno difficoltà con la morfosintassi
- Aumenta la velocità di comunicazione

Un modello basato su transformers come BERT [6] è in grado di fornire una predizione basata sulle relazioni contestuali tra le parole. PictoBERT propone l'utilizzo di BERT per la predizione del pittogramma successivo, con l'assunzione che un pittogramma predetto utilizzando il contesto (i pittogrammi precedenti) sia più significativo di uno predetto da un modello probabilistico o sequenziale che non ve ne tiene conto. Per adattare BERT all'utilizzo di pittogrammi ciascuno di essi viene considerato come concetto e associato ad un word-sense (il significato contestualizzato di una parola) di WordNet, uno dei database lessicali più rinomati e disponibili gratuitamente [7]. Viene posto quindi come obiettivo la predizione del prossimo word-sense, che permetterà di identificare un pittogramma.

Per l'addestramento è stato utilizzato SemCHILDES (Semantic CHILDES), un dataset generato annotando automaticamente il corpus CHILDES (Child Language Data Exchange System [8] con i word-sense corrispondenti. Va menzionato che SemCHILDES non è un dataset di qualità Gold, le trascrizioni delle conversazioni tra bambini e supervisori non fanno distinzione tra frasi espresse dall'uno o dall'altro, inoltre molte delle frasi non sono significative in quanto poteva capitare che suddetti bambini non rispondessero coerentemente alle domande.
Il modello è stato quindi validato confrontandolo con modelli statistici utilizzando la Perplexity come metrica di giudizio, che indica il livello di comprensione del linguaggio in un corpus, ottenendo prestazioni di gran lunga maggiori. In seguito, il modello è stato validato anche da degli esperti del dominio, richiedendo a questi di scrivere delle frasi comunemente utilizzate con i pazienti. Tali frasi sono state poi utilizzate per calcolare la TOP-N metric, che calcola la probabilità che il pittogramma corretto rientri nei primi N predetti dal modello. Dai risultati PictoBERT ha ottenuto risultati sempre superiori agli altri modelli, anche dopo che il suo vocabolario è stato modificato per utilizzare unicamente i pittogrammi ARASAAC.

1.3 Stack Web

Negli ultimi anni, si è verificato un significativo cambiamento nel campo dello sviluppo di applicazioni web verso approcci più efficienti e ottimizzati. Le tradizionali architetture di sviluppo web, come lo stack LAMP (Linux, Apache, MySQL, PHP), stanno lasciando spazio a soluzioni più moderne e versatili. Una di queste soluzioni che ha guadagnato notevole consenso è lo stack MERN, acronimo di MongoDB, Express.js, React.js e Node.js.

1.3.1 MERN

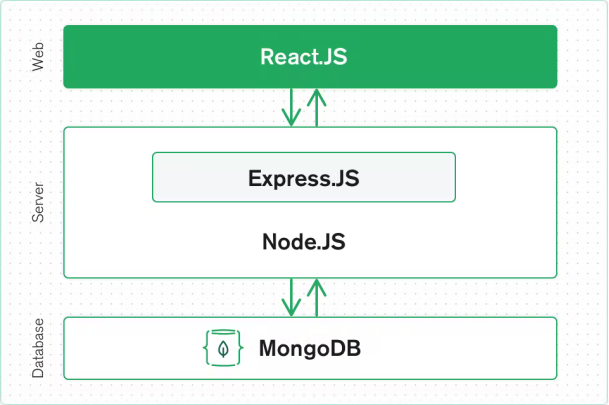

Figura 1.1: Rappresentazione della logica dello stack MERN
Preso da MongoDB

Lo stack MERN [9] è una collezione di tecnologie robuste che consente ai programmatori di costruire applicazioni web full-stack. MERN è diventato sempre più popolare soprattutto grazie alla posibilità di utilizzare JavaScript per l'intero processo di sviluppo, riducendo la curva di apprendimento necessaria per il suo utilizzo. Difatti questo stack è composto da quattro tecnologie principali, tutte basate su JavaScript:

- MongoDB, un database NoSQL che consente di creare file JSON direttamente dal frontend, per poi spedirli al backend dove verranno processati e memorizzati direttamente su MongoDB. Tale approccio garantisce più flessibilità e adattabilità rispetto a un approccio tradizionale con tabelle e righe.
- Express.js, utilizzato come framework server-side, si auto-etichetta come "fast, unopinionated, minimalist web framework for Node.js" [10] e permette la creazione di robuste API lato server.
- React.js, utilizzato per la componente frontend, è un framework sviluppato da Facebook per la creazione di Interfacce Utente (UI). Permette di costruire interfacce complesse e riutilizzabili sotto forma di componenti, e grazie a un DOM virtuale permette la creazione di un'interfaccia reattiva e fluida nell'utilizzo.
- Node.js, un ambiente di runtime JavaScript che permette allo sviluppatore di eseguire codice JavaScript sul server. Node.js è noto per le sue alte

prestazioni e scalabilità ed essendo distribuito con npm (node package manager) garantisce l'accesso ad un'ampia selezione di moduli e librerie.

1.3.2 Serverless

L'approccio tradizionale allo sviluppo di applicazioni web prevede un server sul quale viene eseguito un applicativo e del quale è lo sviluppatore o l'azienda il responsabile, ciò porta con sé una serie di svantaggi:

- Il server rimane in esecuzione anche nei momenti in cui non vi sono richieste, necessitando di una continua spesa anche quando non viene utilizzato
- Lo sviluppatore/azienda è responsabile del mantenimento del server e relative risorse
- Lo sviluppatore/azienda è incaricato di garantire la sicurezza del server
- Il server andrà scalato parallelamente all'utilizzo da parte degli utenti

Tutti questi svantaggi possono essere difficili da gestire per piccole compagnie o sviluppatori autonomi, costringendoli a focalizzarsi su task diversi dallo sviluppo e mantenimento degli effettivi applicativi. Nel caso di aziende grandi vi è un apposito team responsabile, ma anche in questi casi tale processo di supporto alle infrastrutture può rallentare lo sviluppo effettivo.

Lo stack Serverless si pone come una collezione di tecnologie che permette agli sviluppatori di concentrarsi sullo sviluppo, mantenimento e distribuzione delle applicazioni senza doversi preoccupare dell'infrastruttura sottostante.
Il fulcro dello stack Serverless è il Function-as-a-Service (FaaS), che consente ai programmatori di scrivere e distribuire funzioni attivate da eventi o richieste specifiche che, una volta attivate, verranno inoltrate ad un fornitore FaaS, come AWS Lambda, Azure Functions e Google Cloud Functions. Sarà quindi compito del fornitore allocare dinamicamente le risorse e addebitare un costo proporzionale a quelle utilizzate, sollevando i programmatori dall'onere della gestione dei server.

1.4 App Mobile

Negli ultimi anni, con l'avvento dei dispositivi mobile sempre più accessibili e performanti, la richiesta di applicazioni mobile ha avuto una crescita senza precedenti, portando alla nascita di vari framework per semplificarne lo sviluppo.

1.4.1 React Native

React Native [11] è un framework open source per lo sviluppo di applicazioni mobile cross-platform che punta su un approccio "Learn once, write anywhere" o "Impara una volta, scrivi ovunque", facendone d questo il suo motto. Lo sviluppatore può concentrarsi sullo sviluppo di componenti dell'interfaccia in React, lasciando al framework il compito di tradurre i componenti in base alla piattaforma di destinazione attraverso delle API specifiche. Sfruttando

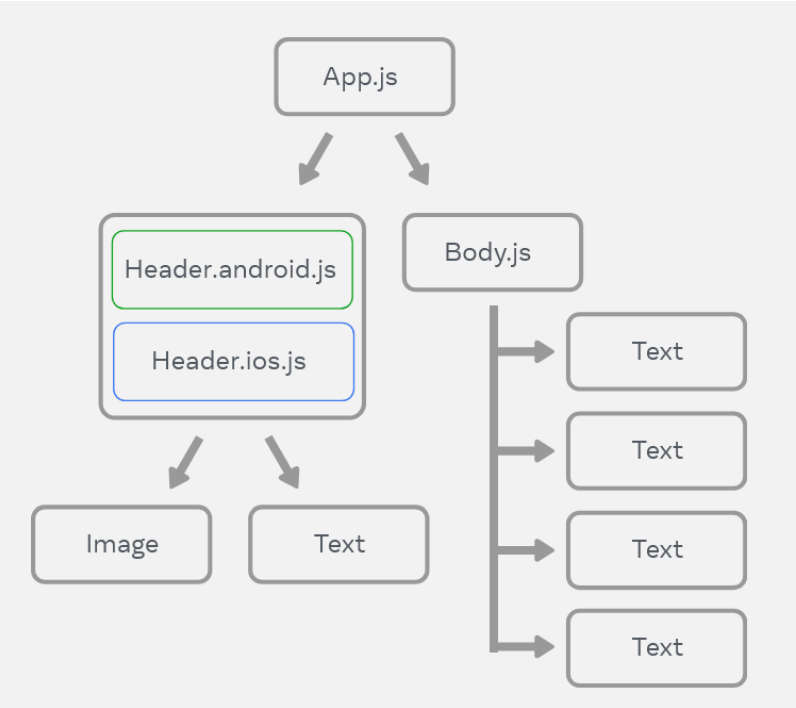

Figura 1.2: Rappresentazione della logica di funzionamento delle API React Native
Preso da React Native

le capacità di rendering native della piattaforma sottostante, le componenti generate avranno quindi prestazioni quasi native, garantendo animazioni fluide, interfacce reattive e tempi di caricamento veloci.
Ulteriore vantaggio di React Native è l'hot-reloading, che consente ai programmatori di vedere in tempo reale le modifiche apportate al codice direttamente nell'applicazione, velocizzando così il processo di sviluppo e debug. Tutti i vantaggi descritti in precedenza, uniti ad un'ampia community ed un ricco ecosistema di librerie, permettono una prototipazione rapida, sviluppo efficiente e semplicità nell'implementazione sia di funzionalità basilari che complesse rendendolo un'ottima scelta .

1.4.2 Flutter

Alternativa a React native per lo sviluppo di applicazioni cross-platform è Flutter, framework open source realizzato da Google in linguaggio Dart. A differenza di React Native il rendering è diretto, ossia non si appoggia alle API native della piattaforma per effettuare il rendering degli elementi ma lo effettua direttamente, garantendo delle interfacce uniformi su tutte le piattaforme. Ciò permette di ottenere in generale delle performance superiori a React Native, essendo però più recente ha una community inferiore e la quantità di librerie attualmente è più limitata.

Capitolo 2

Design

Questo Capitolo descrive la fase di design del progetto, e in particolare le sottofasi di scoping e planning. La prima, riportata in Sezione 2.1, documenta gli incontri con gli esperti di dominio e l'analisi delle alternative sul mercato, che hanno aiutato nella definizione del contesto e delle caratteristiche principali dell'applicativo, denominato da questo punto in poi **PictoAI**. La seconda, in Sezione 2.2, descrive le scelte architetturali che sono state prese in vista della fase implementativa.

2.1 Scoping

2.1.1 Incontri con gli esperti

La fase di scoping è stata svolta tramite una serie di colloqui iniziali con la Prof.ssa. Carbonaro, il Prof. Moro e il Dott. Frisoni, il cui scopo principale è stato quello di introdurre l'argomento e i potenziali requisiti di progetto. In seguito sono stati poi organizzati quattro incontri differenti con gli esperti di dominio, in modo da riuscire a definire dettagliatamente requisiti e caratteristiche da includere nell'applicativo. Fatta eccezione per il quarto incontro, che tratta la validazione del risultato finale (Capitolo 4), l'obiettivo di questa fase è stata composta da:

1. Un primo incontro con presentazione di alcune schermate di PictoAI e un elenco di possibili funzionalità da implementare. In preparazione a quest'ultimo sono state studiate soluzioni simili a quelle che il progetto doveva offrire, riportate in Sezione 2.1.2.

2. Un secondo incontro dove sono stati mostrati agli esperti una serie di mockup in modo da ottenere dei feedback, così da iniziare ad effettuare una

prima fase di pianificazione per poi passare allo sviluppo dell'applicativo con idee più chiare e precise.

3. Un terzo incontro in cui è stato svolta una prima fase di testing utilizzando un prototipo approssimativo dell'applicativo. Durante questo incontro, l'applicativo è stato utilizzato da un gruppo più numeroso di esperti in modo da ottenere ulteriori feedback.

4. Infine, il quarto ed ultimo incontro è stato svolto per validare il risultato ottenuto, assicurandosi che rispettasse le Conditions of Satisfaction (elencate in 2.1.4) definite negli incontri precedenti.

Primo incontro

Il primo incontro ha avuto l'obiettivo di ricevere vari feedback in merito alle idee ottenute inizialmente, nonché di effettuare un brainstorming per eventuali possibili funzionalità da implementare all'interno di PictoAI. Il Dott. Simone Borghi—specializzato in Biotecnologie Industriali, ma che nell'ultimo periodo ha lavorato con bambini affetti da autismo o difficoltà nell'apprendimento—si è dimostrato volenteroso ad aiutare nello sviluppo dell'argomento di questa tesi ed ha preso parte all'incontro.

Dal meeting sono derivate alcune supposizioni e idee interessanti, riportate di seguito.

- I bambini sono stati identificati come target principale dell'applicativo.

- Visto il punto precedente, una feature particolarmente utile sarebbe la presenza di un "companion" (mascotte e aiutante) che agisca in maniera appropriata alle azioni effettuate all'interno dell'app, stimolando in questo modo l'attenzione del bambino mentre la utilizza.

- Il sistema dovrebbe operare seguendo una sorta di routine, concentrandosi su tematiche differenti in giornate differenti.

- L'esperienza deve essere simil-videogame, con la possibilità di giocare mediante i pittogrammi.

- È necessario associare uno stimolo vocale a quello visivo in seguito all'input dell'utente (e.g. una voce deve pronunciare il pittogramma alla sua pressione).

Secondo incontro

In seguito al primo incontro, è stato possibile realizzare una serie di mockup e funzionalità da implementare per le varie pagine, descritti in dettaglio nella Sezione 2.1.3. Durante il secondo incontro, questi modelli sono stati poi validati, in modo da poter procedere con le fasi successive di planning e sviluppo dell'applicativo.

Terzo incontro

Il terzo incontro ha visto la presenza di un numero maggiore di esperti del dominio, con la partecipazione delle Dott.sse Angela Casoria e Irene Bitassi, entrambe psicologhe specializzate in bambini affetti da autismo. In preparazione all'incontro, è stato realizzato un prototipo di PictoAI con alcune funzionalità di base, in modo da poter offrire una versione più concreta e interattiva di ciò che sarebbe stato il prodotto finale. Dall'analisi effettuata sul prototipo, sono emerse una serie di osservazioni:

- L'app deve offrire la possibilità di personalizzare i pittogrammi. Questa personalizzazione infatti aiuta ad agevolare l'utente, offrendogli la possibilità di caricare immagini a lui/lei familiari, come una foto dell'oggetto rappresentato nel pittogramma.
- Alcuni pittogrammi fondamentali devono essere facilmente accessibili, come quelli rappresentanti i bisogni primari (come bagno, fame, sete) o quelli utilizzati più frequentemente.
- Deve essere possibile creare una sorta di diario su cui il bambino può annotare, senza alcuna necessità di supervisione, le esperienze passate, in modo da poterne discutere in seguito.
- I colori dei pittogrammi devono essere significativi, rispecchiando ad esempio la categoria di appartenenza.
- È necessario limitare i pittogrammi visualizzati simultaneamente ad un massimo di 9 per schermata.

2.1.2 Observing

Per ottenere una panoramica sul contesto dell'applicativo, il primo incontro è stato preceduto da uno studio delle feature presenti su simili alternative già presenti sul mercato. Questa fase ha portato alla realizzazione di una presentazione, che è stata analizzata durante l'incontro. L'elenco seguente riporta le caratteristiche di ogni applicativo selezionato.

- **Proloquo2Go AAC** [12], applicativo che espone una classica board di pittogrammi. Alcune sue peculiarità sono:
  - possibilità di utilizzare l'app come una sorta di tastiera virtuale in modalità Picture-in-Picture (PiP), in contemporanea all'uso di altre applicazioni;
  - suddivisione dei pittogrammi in categorie;
  - presenza, per ogni pittogramma, di un bordo colorato che ne indica la categoria;
  - insegnamento progressivo: il vocabolario di pittogrammi viene inizialmente limitato, per poi ampliarsi gradualmente a mano a mano che l'utente prende confidenza con i pittogrammi già presenti;
  - possibilità di personalizzare la board di pittogrammi predefinita, impostando le parole preferite dall'utente;
  - possibilità di modificare i pittogrammi esistenti.
- **TouchChat** [13], applicazione che offre una predizione dei pittogrammi successivi, senza l'utilizzo di AI.
- **The Grid 3** [14], applicativo che offre due variazioni del vocabolario, adatte rispettivamente ad un livello base e avanzato dell'utente. Altra peculiarità è la possibilità di utilizzo mediante eye tracking.
- **LAMP Words for Life** [15], dallo stesso sviluppatore di TouchChat, si concentra sull'insegnamento dei movimenti necessari ad esprimere un concetto. Nell'app, i pittogrammi vanno posizionati secondo un determinato pattern in base al concetto che rappresentano, ponendo il focus sui movimenti che lo esprimono.
- **SIMCAA** [16], app open source che ha come peculiarità la possibilità di salvare le sequenze di pittogrammi generate in "storie", che possono essere poi consultate o modificate in un secondo momento.
- **Leeloo AAC** [17], applicativo per bambini che ha attirato l'attenzione per il suo design molto interessante (Figura 2.1) e per la possibilità di leggere le frasi generate con un sistema di Text-To-Speech (TTS), che permette la selezione di varie voci.
- **CoughDrop AAC** [18], app simile agli altri, ma che pone un maggior focus sul supervisore, fornendo varie statistiche dell'utilizzo da parte degli utenti.

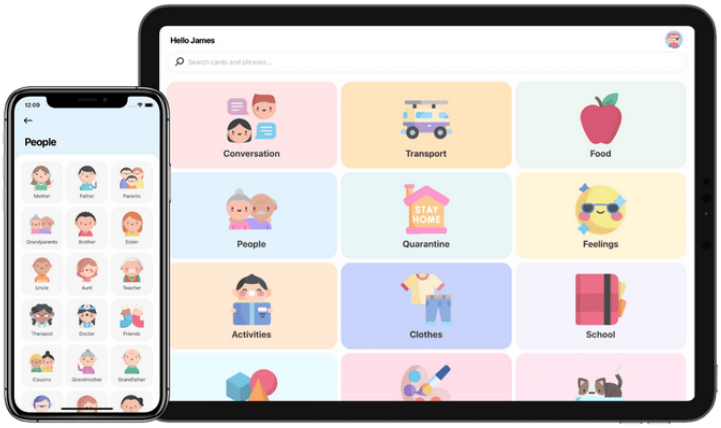

Figura 2.1: Esempio di schermate dell'app Leeloo AAC
Preso da Play Store

- **Spoken** [19], applicativo che cerca di "imparare" informazioni sull'utente, ponendo periodicamente delle domande come "Hai qualche animale domestico?", in modo da ampliare la propria gamma di informazioni per poi utilizzarla in seguito durante l'insegnamento.

- **Leo AAC & Autism Speech Aide** [20], applicazione che pone particolare focus sulla personalizzazione, permettendo ad esempio di registrare la propria voce per i singoli pittogrammi, in sostituzione al lettore TTS predefinito

- **Aac Talking Tabs** [21], applicativo che propone vari giochi per imparare ad utilizzare i pittogrammi, come quello di completare una frase col pittogramma corretto. Altra caratteristica interessante è un catalogo di libri consultabili sotto forma di pittogrammi.

2.1.3 Mockup

In preparazione al primo incontro sono stati realizzati alcuni mockup di PictoAI. Si è scelto di fornire già una rappresentazione dettagliata, seppur aperta a modifiche, dell'interfaccia che sarebbe stata implementata, in modo da avere un'idea precisa sulla soluzione da sviluppare in seguito all'incontro.

L'idea iniziale di PictoAI prevede tre componenti principali, come mostrato in Figura 2.2. Ciascun componente, o Sezione dell'applicativo, avrebbe fornito uno strumento per l'insegnamento o il miglioramento delle capacità comunicative dell'utente secondo approcci diversi. La *modalità gioco*, ad esempio, propone una serie di mini-giochi, generati col supporto dell'intelligenza artificiale, che

puntano a coinvolgere l'utente in un'esperienza di insegnamento interattiva, personalizzata in base alle proprie preferenze/scelte e che fornisca un feedback stimolante e positivo in seguito alle risposte corrette. Per permettere all'utente di concentrarsi su una certa categoria di pittogrammi, durante l'insegnamento è prevista anche la possibilità di selezionare una categoria specifica da utilizzare per la generazione dei giochi.

Figura 2.2: Mockup della home di PictoAI

Figura 2.3: Mockup della pagina "Giochiamo"

Per ciascuno dei giochi, rappresentati in Figura 2.3, è stato realizzato un mockup della relativa schermata accompagnato da una sintesi del suo funzionamento, riportata di seguito.

- **A parole tue**
  In questa modalità viene mostrato un video che l'utente deve descrivere utilizzando i pittogrammi. Una volta scelta e confermata la sequenza di pittogrammi, essi vengono a loro volta utilizzati per generare un video, consentendo all'utente di osservare le differenze tra il video iniziale e quello generato dalle proprie scelte. Entrambi i video verrebbero generati tramite IA.

- **Quiz**
  Modalità in cui la mascotte pone varie domande, in certi casi anche frasi da completare, e l'utente deve rispondere mediante pittogrammi. Sia i quesiti che i pittogrammi proposti verrebbero generati tramite IA.

- **Cosa senti?**
  Mini-gioco in cui la mascotte legge una frase, generata tramite IA, menzionando un elemento che l'utente deve indovinare mediante pittogrammi.

- **Che cos'è?**
  L'utente deve indovinare un pittogramma, a partire da un'immagine e un prompt entrambi generati via IA.

- **Crea una storia**
  In questa modalità, vengono proposti all'utente una serie di soggetti, predicati e oggetti. Una volta selezionati, viene generato un filmato corrispondente ai pittogrammi selezionati.

La pagina "Leggiamo", mostrata in Figura 2.4, permetterebbe di consultare un libro rappresentato sotto forma di pittogrammi, in autonomia o mediante TTS, evidenziando ogni pittogramma mentre viene letto dall'assistente vocale. Alcuni libri potrebbero essere forniti insieme all'applicativo, con la possibilità per l'utente di crearne di nuovi a partire da un file testuale, che verrebbe elaborato da un'IA per la generazione dei relativi pittogrammi.

Infine, la modalità "Parliamo", in Figura 2.5, permette all'utente di creare frasi col supporto dell'IA, facilitando la comunicazione con le persone nelle vicinanze. L'IA in questo caso servirebbe a suggerire i pittogrammi che l'utente potrebbe voler utilizzare, in base alla categoria selezionata e ai pittogrammi inseriti in precedenza.

Figura 2.4: Mockup della pagina "Leggiamo"

Figura 2.5: Mockup della pagina "Parliamo"

2.1.4 Conditions of Satisfaction

- Realizzazione di un applicativo mobile che permetta di integrare componenti di intelligenza artificiale per supportare l'apprendimento di persone con maggiori difficoltà
- L'applicativo generato deve poter essere traslato facilmente in sito web
- L'applicativo deve essere abbastanza semplice da poter essere utilizzato da persone con difficoltà cognitive mediante l'utilizzo di pittogrammi

- L'applicativo deve essere adatto all'utilizzo da parte di bambini
- Associazione di uno stimolo vocale oltre a quello visivo durante l'utilizzo dell'applicativo
- Realizzazione di una componente simil-videogame che sfrutti i pittogrammi per scopi educativi
- Presenza di un "companion" che stimoli l'utente fornendo un feedback alle sue azioni
- Possibilità di concentrare l'insegnamento su tematiche specifiche
- Possibilità di personalizzare i pittogrammi in maniera semplice
- Presenza di pittogrami accessibili facilmente durante la scrittura
- Realizzazione di una componente che permetta di visualizzare e scorrere libri sotto forma di pittogrammi
- Colorazione dei pittogrammi significativa
- Presenza di una componente che permetta all'utente di utilizzare i pittogrammi per comunicare con altre persone
- Presenza di una componente che permetta all'utente di salvare sequenza di pittogrammi sotto forma di diario
- Permanenza dei dati ricavati durante utilizzo dell'applicativo

2.2 Planning

La fase di planning del progetto è stata affrontata sin dai primi colloqui con la Prof.ssa Carbonaro e il Prof. Moro: avendo a disposizione una quantità limitata di tempo, si è cercato da subito di elaborare una soluzione che fosse efficace ma flessibile, in modo da non riscontrare difficoltà nell'introdurre cambiamenti in seguito agli incontri con gli esperti. L'idea di partenza era quella di realizzare un applicativo destinato ad un utilizzo prevalentemente web, pertanto lo sviluppo si era inizialmente concentrato sull'implementazione di suddetta parte. Durante il terzo incontro è però emersa la necessità di fornire agli utenti uno strumento sempre disponibile tramite smartphone, per aiutarli con la comunicazione. L'obiettivo di questa tesi è diventato quindi lo sviluppo di un applicativo mobile, che in futuro possa essere trasposto su una controparte web. Tenendo conto della necessità di adattare l'insegnamento alle

esigenze dell'utente, garantendo la conservazione dei dati utente durante la migrazione tra dispositivi e consentire una possibile raccolta futura di statistiche per migliorare sia l'esperienza dell'utente che la qualità degli algoritmi di intelligenza artificiale, sarebbe opportuno prevedere la presenza di un Database e sviluppare una componente di Backend che offra API per l'interazione con quest'ultimo.

Un altro importante requisito derivato dagli incontri, è la necessità di limitare o possibilmente evitare ogni spesa in infrastrutture, in quanto molte delle organizzazioni a supporto di persone con difficoltà nell'apprendimento sono non profit e dispongono di budget molto limitati. L'architettura di sistema che ha consentito il rispetto di questo requisito è dettagliata nella Sezione 2.2.1, mentre le tecnologie da essa utilizzate sono elencate in Sezione 2.2.2.

2.2.1 Architettura

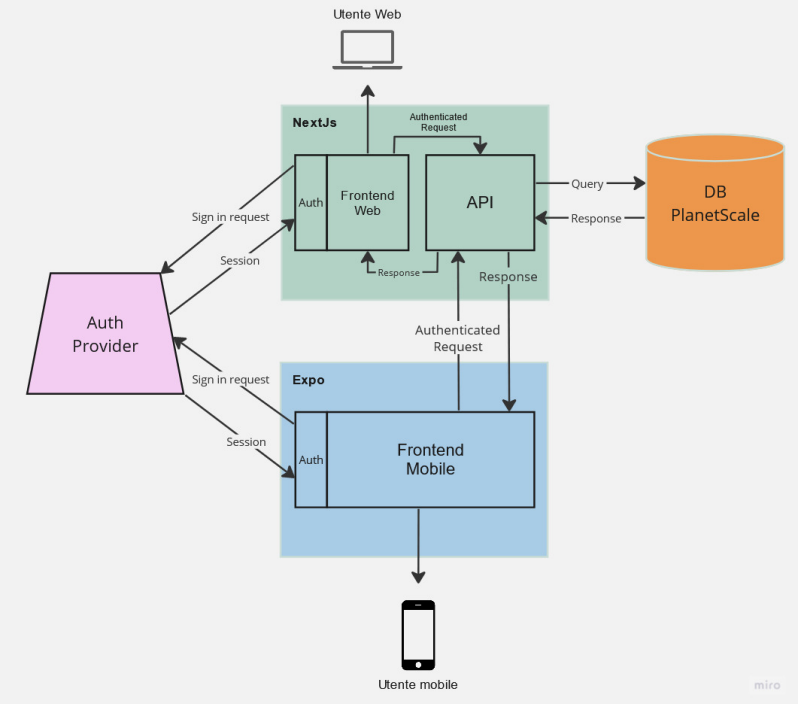

Figura 2.6: Architettura del sistema in fase di Planning

L'architettura del sistema, mostrata in Figura 2.6, prevede:

- Un frontend Web per l'interazione con l'utente e la comunicazione con API in seguito all'autenticazione da un Authentication Provider
- Un frontend Mobile analogo a quello Web ma per dispositivi mobile

- Un Authentication Provider che assicuri l'identità dell'utente
- Delle API Backend che, in seguito ad una richiesta autenticata lato frontend, interagiscano col Database fornendo una risposta
- Un DB per la persistenza dei dati

2.2.2 Tecnologie e servizi utilizzati

Vista l'architettura del sistema, che comprende un database, un set di API web e un'applicativo mobile, in aggiunta alla possibilità di realizzare una web app in futuro, è necessario prestare molta attenzione alla selezione delle tecnologie da utilizzare, in modo che la loro adozione non introduca debito tecnico, ma consenta invece una buona flessibilità in fase implementativa e un'apertura ad eventuali sviluppi futuri.

Data la premessa, la scelta è ricaduta sullo Stack T3 [22], una raccolta di strumenti per la creazione di applicazioni sia mobile che web in linguaggio TypeScript, con l'obiettivo di consentire uno sviluppo rapido, flessibile, totalmente typesafe e un hosting altamente scalabile, ma con zero costi iniziali. Le componenti principali dello stack sono il linguaggio TypeScript, il framework Next.js e le librerie tRPC, Tailwind CSS e Prisma. Altre librerie utilizzate, non facenti parte dello stack, sono riportate di seguito, così come le soluzioni individuate per l'hosting.

TypeScript

L'utilizzo del linguaggio TypeScript [23], che introduce un layer di tipizzazione statica al di sopra di JavaScript, permette di ottenere un feedback in tempo reale da parte dell'editor, che è in grado di offrire suggerimenti ed individuare errori a compile-time, invece che a runtime. Inoltre, l'adozione di TypeScript per tutte le componenti del sistema (backend, app e, in futuro, sito web) riduce notevolmente la complessità della codebase e offre la possibilità di condividere facilmente codice tra le varie parti.

Next.js

Next.js [**?**] è un metaframework per React, che abilita il server-side rendering dei contenuti. Le applicazioni React tradizionali vengono renderizzate lato client, il che potrebbe non essere desiderato in alcune situazioni. Ad esempio, i crawler dei motori di ricerca potrebbero avere difficoltà a indicizzare i contenuti del sito. Feature di maggiore importanza per il progetto è però la possibilità di creare delle API route, ossia rotte che vengono eseguite lato server e restituiscono

informazioni sotto forma di JSON. Grazie a questa funzionalità, un applicativo Next.js è a tutti gli effetti full-stack e non richiede l'utilizzo di altri framework, come Express, per la realizzazione della logica di business lato backend.

tRPC

tRPC [24] è una libreria, alternativa all'utilizzo di uno stile REST o di tecnologie come GraphQL, che permette di creare e consumare un set di API totalmente typesafe, senza la necessità di ricorrere a schema o code generation. Alla base del suo funzionamento vi è il linguaggio TypeScript, che consente di inferire i tipi di input e output di ogni procedura definita lato server, e di fornirli al client in modo che ne sia a conoscenza.

Tailwind CSS

Tailwind CSS [25] è un framework utilizzato per modellare lo stile del proprio sito o applicativo. Fornendo blocchi di costruzione sotto forma di colori predefiniti, spaziatura e altre primitive. Tailwind semplifica la creazione di un'applicazione di bell'aspetto mantenendo una flessibilità pari a quella del normale CSS. Inoltre, grazie al suo approccio inline, Tailwind permette di creare un design senza preoccuparsi di dare un nome alle classi, organizzare i file di stile, o di tutte le questioni non direttamente legate al problema da risolvere.

Prisma

Prisma [26] è un ORM (Object-Relational Mapping) per il linguaggio TypeScript, che permette di interagire in maniera totalmente type safe con i maggiori DBMS sul mercato. Prisma utilizza un proprio DSL (Domain-Specific Language) per la definizione della struttura delle tabelle, applicata al database tramite una semplice CLI, che si occupa anche di generare i tipi di TypeScript per tutte le query possibili.

PNPM

PNPM (Performant NPM) [27] è un package manager per la gestione delle dipendenze di progetti Node.js. A differenza di NPM, PNPM adotta un'approccio diverso per la gestione delle dipendenze: mentre NPM installa i pacchetti in modo centralizzato all'interno di ogni progetto, PNPM adotta una modalità di installazione condivisa. Ciò significa che i pacchetti vengono scaricati una sola volta nel sistema e vengono collegati in modo simbolico ai

progetti che li richiedono. Questo ha permesso di raggiungere un notevole livello di modularità e di riuso del codice all'interno del progetto.

React Native

React Native [11] permette di creare applicazioni cross-platform tramite l'utilizzo di React per la definizione dell'interfaccia utente. Questo framework offre vari target di compilazione, da Android (utilizzato per il progetto) a iOS, web, Windows e anche macOS. Data la possibilità di investire su un unico team e una sola codebase per la realizzazione di app multipiattaforma, è adottato da innumerevoli compagnie, come Meta, Microsoft, Discord e Tesla.

Expo

Expo [28] è un framework che estende React Native, offrendo ulteriori servizi come gli Expo Application Services, una piattaforma cloud che permette di creare bundle di distribuzione senza la necessità di configurare manualmente certificati o di utilizzare complessi strumenti di compilazione. Expo si appoggia all'app Expo Go, che permette di eseguire e fare debugging di un'applicazione con hot-reloading tramite simulatore o dispositivo reale.

Vercel

Vercel [29] è una piattaforma per sviluppatori che fornisce strumenti e infrastruttura per il deployment di applicazioni web, integrandosi in maniera trasparente con servizi come GitHub senza la necessità di alcuna configurazione iniziale. Il suo piano gratuito consente il deployment di un numero illimitato di siti web, a patto di mantenere una banda mensile inferiore ai 100 GB/sito, con la possibilità di passare ad un piano a pagamento qualora il numero di utenti attivi aumenti al punto da rendere necessaria una quantità maggiore di banda.

Planetscale

Planetscale [30] è un database MySQL serverless, che offre ottime performance ed elevati livelli di scaling, con gestione automatica dello sharding. Come Vercel, ha un piano gratuito estremamente generoso - 1 miliardo di letture di righe al mese e 10 milioni di scritture - con la possibilità di effettuare un upgrade a pagamento qualora ciò fosse necessario.

Clerk

Clerk [31] è un sistema di autenticazione e user management per applicativi web e mobile, che si integra con i maggiori provider di SSO (Single Sign-

On) per fornire un servizio facilmente configurabile, altamente scalabile e particolramente economico. Infatti, è totalmente gratuito sotto alla soglia di 5000 utenti attivi.

Capitolo 3

Implementazione

Questo Capitolo descrive la fase di implementazione di PictoAI, suddividendola in ambiente di sviluppo (Sezione 3.1), frontend (Sezione 3.2), backend (Sezione 3.3), database (Sezione 3.4) e deployment (Sezione 3.5).

3.1 Ambiente di sviluppo

L'ambiente di sviluppo di PictoAI è composto dai seguenti servizi/strumenti:

- Visual Studio Code [1]: IDE utilizzato per la scrittura del codice
- Android Studio [2]: IDE utilizzato per la sua funzionalità di Android Virtual Device Manager, che ha permesso di creare dispositivi virtuali Android su cui eseguire l'app in fase di sviluppo. In particolare, è stato configurato per la simulazione di un dispositivo Google Pixel 6 (API 33) come smartphone, e Pixel C (API 33), come tablet
- Smarthphone OnePlus Nord 2 5G: dispositivo utilizzato per il testing mobile dell'applicativo
- Git [3]: distributed version control system utilizzato per il tracciamento delle modifiche ai file del progetto, il ripristino delle versioni precedenti e la gestione dei rami di sviluppo
- GitHub: piattaforma di hosting per repository Git utilizzata per la gestione del codice, il monitoraggio delle modifiche e il rilascio. Link alla repository utilizzata: https://github.com/mcnuggetboii/AAC-Tesi

[1] https://code.visualstudio.com/
[2] https://developer.android.com/studio
[3] https://git-scm.com/

Gli Android Virtual Device sono stati utilizzati per eseguire l'applicativo durante le fasi iniziali dello sviluppo, mentre nelle fasi finali si è ricorso ad un dispositivo fisico, in modo da creare un'esperienza di utilizzo più accurata. Per il backend Next.js, è stato sempre utilizzato un server locale in modalità development fino alla sua release su Vercel, mentre per la sua connessione al database è stata utilizzata una stringa di connessione impostata tramite una variabile d'ambiente.

3.2 Frontend

Lo sviluppo della parte di frontend è stato effettuato mediante React Native e il framework Expo, descritto in Sezione 2.2.2. L'organizzazione dei file del progetto è riportata di seguito:

- app:
  - components: componenti utilizzati nelle pagine
  - hooks: hook utilizzati dai componenti
  - store: soluzione di state management utilizzata nell'applicativo
  - utils: funzioni o porzioni di codice comuni, utilizzate in varie parti dell'applicativo
  - views: pagine visualizzabili dall'utente
  - _layout.tsx: componente React che funge da container per tutte le pagine renderizzate dal router. Quest'ultimo permette di definire le funzioni da eseguire prima di renderizzare la Home page e l'interfacciamento con le librerie utilizzate, selezionando i contenuti che un utente non autenticato può visualizzare, e i componenti presenti in ogni pagina
  - index.tsx: home page dell'applicativo, descritta in 3.2.5
- app.config.ts: file che permette di configurare tutte le impostazioni dell'applicativo come nome, versione, orientazione, icona e altro. Nel campo extra vanno inserite anche le variabili ambientali da usare in fase di rilascio e l'id dell'applicativo fornito da expo
- babel.config.ts: file per la configurazione di Babel [4] e dei relativi plugin
- eas.json: file di configurazione della CLI EAS [5], utilizzata per creare l'apk dell'applicativo, descritto in 3.5.1

[4] https://babeljs.io/
[5] https://github.com/expo/eas-cli

- metro.config.ts: file per la configurazione di Metro [6], bundler JavaScript che raggruppa e ottimizza il codice sorgente e le risorse per una distribuzione più efficiente su dispositivi mobili

- package.json: file utilizzato per gestire le dipendenze di pnpm e gli script del progetto

- tailwind.config.ts: file per la configurazione di Tailwind, permette di definire dei nomi personalizzati da assegnare alle classi di Tailwind

- tsconfig.json: file che specifica i file sorgente e le opzioni del compilatore

3.2.1 Componenti

Un componente React è un blocco di codice che incapsula la logica e la rappresentazione visuale di una parte specifica dell'interfaccia utente. Ciascun componente del progetto è scritto in linguaggio Typescript (.tsx) e definisce l'interfaccia da renderizzare in base alle proprietà (props) e allo stato. Più componenti possono essere utilizzati insieme per creare elementi complessi dell'interfaccia utente, mantenendo al contempo modularità e manutenibilità.

Companion

Companion è l'implementazione del concetto di mascotte definito in 2.1.1 e rappresentato in Figura 3.1. Il componente viene renderizzato in una delle posizioni sullo schermo (centro o destra), tramite un'immagine rappresentante la mascotte. A quest'ultima è annessa una nuvola in stile fumetto (anch'essa visualizzabile in alcune posizioni predefinite) che viene mostrata a schermo quando, da una qualsiasi pagina, viene richiesto un feedback vocale per associare alle parole pronunciate un testo scritto. La visibilità del Companion e delle sue componenti è definita da alcune variabili globali, condivise in tutto l'applicativo e descritte in Sezione 3.2.3. Una pressione da parte dell'utente sul Companion o sulla nuvola viene ignorata, indirizzando l'evento all'elemento sottostante. In tutte le pagine in cui viene visualizzato il Companion sono presenti anche delle icone nell'angolo inferiore sinistro dello schermo, tra cui una nuvola testuale che, se premuta, rimuove o ripristina la presenza della nuvola durante la pronuncia del testo

Al momento della stesura di questa tesi è stata utilizzata un'immagine statica [7], per la rappresentazione del Companion. In futuro, questa potrebbe

[6] https://facebook.github.io/metro/

[7] https://www.cleanpng.com/png-gaoqiaozhen-teacher-education-skill-teacher-123819/

Figura 3.1: Componente Companion nell'applicativo

venire sostituita da una Figura animata e interattiva, realizzata con l'aiuto di un esperto di grafica.

PictogramCard

PictogramCard (mostrato in Figura 3.2) è il componente che permette la visualizzazione di un pittogramma con eventuale testo associato. Questo componente è adattabile a ciascuna pagina mediante una serie di prop:

- `pictogram` (`Pictogram | undefined`): indica l'elemento di tipo `Pictogram` associato al componente, da questo verranno poi ricavati l'ID dell'immagine da mostrare ed il relativo testo
- `text` (`string | undefined`): se impostato permette di sovrascrivere il testo base del pittogramma con quello fornito
- `noCaption` (`boolean | undefined`): se impostato a `true`, indica di non visualizzare il testo, ma unicamente l'immagine del pittogramma
- `full` (`boolean | undefined`): utilizzato per creare griglie di pittogrammi, se impostato a `true` indica che il pittogramma deve occupare tutto lo spazio assegnato
- `horizontal` (`boolean | undefined`): se impostato a `true` indica che il pittogramma deve essere visualizzato in modalità orizzontale, mostrando l'immagine a destra del testo
- `radius` (`number | undefined`): valore numerico che indica un eventuale curvatura nel bordo del pittogramma

- `bgcolor` (`string | undefined`): colore in formato esadecimale da sovrascrivere a quello di base del pittogramma
- `onPress` (`(...args: any)` => void): funzione richiamata alla pressione di un pittogramma
- `args` (`any | undefined`): eventuali argomenti da passare alla funzione `onPress`
- `border` (`string | undefined`): colore in formato esadecimale di un eventuale bordo

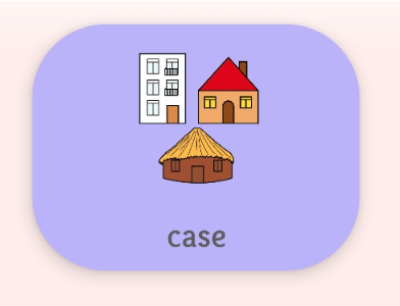

Figura 3.2: Esempio di utilizzo del componente PictogramCard

PictogramSearchFlatlist

Componente utilizzato per la ricerca di pittogrammi (mostrato in Figura 3.3), composto da una barra di ricerca, una FlatList [8] (componente base di React Native ottimizzato per la visualizzazione di un ampio numero di elementi caricati in modalità lazy), e da alcune prop:

- `defaultText` (`string`): testo visualizzato prima della ricerca o se `defaultData` è vuoto
- `defaultData` (`Pictogram[] | undefined`): array di pittogrammi da utilizzare quando non sono impostati dei criteri di ricerca, solitamente usato per mostrare i pittogrammi preferiti dall'utente
- `backgroundColor` (`string | undefined`): colore, in formato esadecimale, dello sfondo
- `inputColor` (`string | undefined`): colore, in formato esadecimale, specifico del testo all'interno della barra di ricerca

[8] https://reactnative.dev/docs/flatlist

- `onSelect ((el: Pictogram) => void)`: funzione eseguita alla pressione di un pittogramma

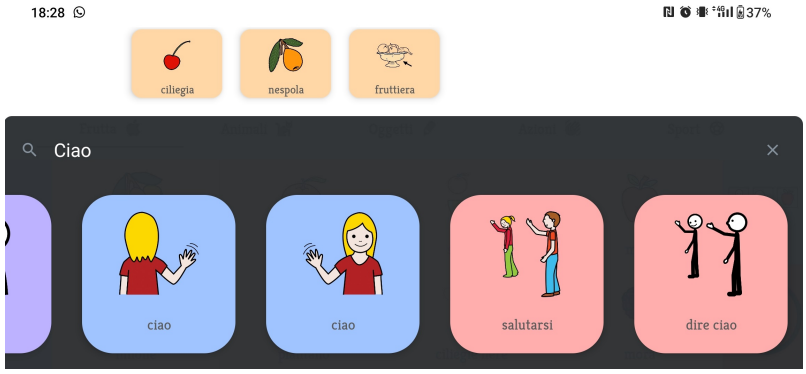

Figura 3.3: Componente modale che utilizza un PictogramSearchFlatlist

PictogramCustomizationModal

Modale utilizzato per la personalizzazione di pittogrammi. Viene suddiviso in tre schermate, accessibili tramite i pulsanti mostrati in Figura 3.4. La schermata principale è composta da una PictogramCard che raffigura il pittogramma personalizzato. Premendo su quest'ultima, è possibile selezionare un pittogramma standard (mediante PictogramSearchFlatlist) che verrà poi sostituito da quello inizialmente creato. Gli altri elementi all'interno della schermata, utilizzati per personalizzare il pittogramma, sono: una textbox per impostare il testo, un pulsante per aggiungere un'immagine personalizzata, un pulsante per impostare un colore pesonalizzato al pittogramma e un pulsante per associare delle categorie (per i pittogrammi creati ex novo) fino ad un massimo di 5. Va però fatto notare che è possibile creare due tipologie di pittogrammi personalizzati:

- Un pittogramma nuovo, senza alcuna associazione a quelli esistenti, che potrà venire utilizzato nelle altre pagine dal momento della sua creazione
- Un pittogramma associato ad uno già presente, che sostituirà tutte le occorrenze di quello vecchio

3.2.2 Hooks

useHuggingFace

Use Hugging Face contiene tutte le chiamate verso Hugging Face [32], servizio che offre una vasta gamma di modelli pre-addestrati per NLP e fornisce

Figura 3.4: PictogramCustomizationModal, modale per la personalizzazione dei pittogrammi

agli sviluppatori la possibilità di condividere i propri. Insieme al Dott. Frisoni è stato deciso che la componente AI verrà condivisa tramite quest'ultimo, rappresentando di fatto il punto di integrazione delle varie componentistiche di IA con l'applicativo.

Al momento della stesura della tesi i modelli di IA non erano ancora completi, pertanto tutte le chiamate effettuate da useHuggingFace non avevano un URL di destinazione impostato, ma servivano da Proof-Of-Concept per dimostrare come, una volta pubblicati i modelli, i dati venivano recuperati ed elaborati. In caso di incongruenze con le versioni finali delle risposte da parte dei modelli è necessario modificare le interfacce dichiarate nel file (ParsedFileResponse, GeneratedWhatsItGameResponse e PredictedPictogramsResponse) con le loro nuove varianti ed eventualmente effettuando un parsing per ottenere i dati che sono rilevanti per l'applicativo. Per effettuare queste chiamate è necessario impostare una variabile ambientale chiamata **HUGGINGFACE_BEARER** che rappresenta il token di autorizzazione fornito da Hugging Face.

Le chiamate vengono effettuate mediante la libreria Axios, che semplifica il processo di gestione delle richieste verso API REST, gestito in modo da poter rigenerare una richiesta, fino ad un massimo di tre volte, con un delay pari a tre secondi in caso di mancata risposta. Le chiamate effettuate sono le seguenti:

- getPictogramsFromFile: chiamata per inviare un file testuale da convertire in pittogrammi
- requestWhatsItGame: chiamata per generare una partita di "Che cos'è?", fornendo eventualmente una categoria
- predictPictograms: chiamata per ottenere le predizioni dei prossimi pittogrammi da digitare, fornendo eventualmente in input quelli precedenti, quelli correnti, una categoria selezionata e delle impostazioni sulla complessità dei pittogrammi richiesta

Da notare che le chiamate potrebbero fornire anche l'ID dell'utente, in modo che il modello possa connettersi ed utilizzare eventuali dati e statistiche sull'utente presenti già nel DB.

3.2.3 Store

Su React, lo stato rappresenta un dato dinamico che può cambiare a runtime durante l'esecuzione dell'applicativo. Lo State Management rappresenta le pratiche di gestione dello stato e può diventare complesso quando questo deve essere condiviso tra vari componenti. Per risolvere il problema, vengono fornite delle librerie come Zustand [9], utilizzato all'interno di questo applicativo, per la centralizzazione dello stato di un'applicazione all'interno di uno "store" globale, semplificando di conseguenza la condivisione di dati tra componenti e fornendo uno strumento per gestire l'accesso e gli aggiornamenti allo stato.

Companion State

Componente dello store utilizzata per la gestione della mascotte, descritta in Sezione 3.2.1. Dato che quest'ultima appare in diverse pagine, il Companion State offre funzioni per gestirne la posizione e la nuvola di testo, implementa il TTS e permette di accedere al testo che è in fase di lettura. L'interfaccia esposta da Companion State è la seguente:

```
interface CompanionState {
  isVisible: boolean;
  currentText: string;
  position: string;
  bubblePosition: string;
  volumeOn: boolean;
  bubbleOn: boolean;
  speak: (
    text: string,
    bubblePosition?: string,
    onBoundary?: (e: any) => void, // Define some code to execute at each space
    onDone?: () => void, // Define some code to execute when the reading is done
  ) => Promise<void>;
  resume: () => void;
  pause: () => void;
  resetSpeech: () => Promise<void>; // Stop reading and reset all variable to
  ↪  default
  changeVolume: () => Promise<void>;
  changeBubble: () => void;
  setPosition: (newPosition: string) => void;
  setVisible: (value: boolean) => void;
  hideAll: () => void;
  showAll: () => void;
}
```

[9] https://zustand-demo.pmnd.rs/

Input State

Componente dello store necessaria per utilizzare la pagina "Parliamo", descritta in Sezione 3.2.8. Genera un input che viene propagato tra le varie pagine o tra i vari componenti e, attraverso le funzioni esposte, permette di generare una richiesta definita da un ID e un comando, per ottenere in risposta una sequenza di Pittogrammi. L'interfaccia esposta da Input State è la seguente:

```
    interface inputState {
      id: string | undefined;
      command: string | undefined;
      inputPictograms: string[] | undefined;
      requestCompleted: boolean;
      args: any;
      inputRequest: (id: string, command: string, args?: any) => void;
      setInput: (reqID: string, inputPictograms: string[]) => void;
      clear: () => void;
    }
```

È possibile fornire anche dei pittogrammi in input, nel caso questi debbano essere modificati.

Pictogram State

Una delle componenti più importanti dello Store, si occupa della gestione dei Pittogrammi. Permette di accedere alle funzioni per la lettura e modifica dei pittogrammi e di gestire i pittogrammi preferiti. L'interfaccia esposta da Pictogram State è la seguente:

```
interface PictogramState {
  pictograms: Pictogram[];
  favourites: string[];
  customPictograms: CustomPictogram[];
  showAdjectives: boolean;
  load: () => Promise<void>;
  save: () => Promise<void>;
  setShowAdjectives: (value: boolean) => Promise<void>;
  getPictogram: (id: string) => Pictogram | undefined;
  getPictograms: (ids: string[]) => Pictogram[];
  getTextFromPictogram: (pictogram: Pictogram) => string | undefined;
  getPictogramByText: (text: string) => Pictogram[];
  getFavouritePictograms: () => Pictogram[];
  getPictogramFromCustom: (
    customPictogram: CustomPictogram,
  ) => Pictogram | undefined;
  getCustomPictograms: () => Pictogram[];
  addFavourite: (id: string) => Promise<boolean>;
  removeFavourite: (id: string) => Promise<boolean>;
  addCustomPictogram: (
    oldId?: string,
```

```
    text?: string,
    image?: string,
    tags?: string[],
  ) => Promise<boolean>;
  removeCustomPictogram: (id: string) => Promise<boolean>;
}
```

In seguito una descrizione più dettagliata di alcune delle funzioni più rilevanti:

- showAdjectives: valore booleano che, se impostato a `true`, indica alla componente AI di generare pittogrammi più avanzati, comprensivi di aggettivo
- showColors: valore booleano che, se impostato a `true`, permette di visualizzare i pittogrammi con lo sfondo associato alla categoria
- bigMode: valore booleano che, se impostato a `true`, indica la necessità di visualizzare i pittogrammi in una forma più grande e leggibile quando possibile, ad es. nella pagina *Parliamo*
- getPictogram: fornisce un Pittogramma, predefinito o personalizzato
- getPictogramByText: ricerca i pittogrammi contenenti una stringa di testo fornita come parametro. Il risultato viene ordinato in base alla distanza di Levenshtein [10] tra la parola ricercata e i pittogrammi. Si è deciso di utilizzare questa misura solamente per i pittogrammi che contengono la parola scelta per motivi di performance e per semplificare la ricerca. Si è notato, durante lo sviluppo, che applicare suddetta metrica su tutti i pittogrammi rallentava notevolmente la velocità di caricamento dei risultati.

Book State

Componente dello store adibita alla gestione dei libri. Le funzioni e le variabili esposte permettono di leggere, aggiungere o rimuovere i libri generati dall'utente e visualizzati nella pagina **Leggiamo**. L'interfaccia esposta da Book State è la seguente:

```
interface BookState {
  customBooks: Book[];
  readingSettings: ReadingSettings;
  load: () => Promise<void>;
  save: () => Promise<void>;
  getBook: (id: string) => Book | undefined;
  addBook: (book: Book) => Promise<boolean>;
```

[10] https://it.wikipedia.org/wiki/Distanza_di_Levenshtein

```
8     removeBook: (id: string) => Promise<boolean>;
9   }
```

3.2.4 Login

L'implementazione del processo di login è stato effettuata utilizzando Clerk, descritto in Sezione 2.2.2. Per la sua configurazione è necessario definire un container nel file _layout.tsx, all'interno del quale collocare tutti i componenti. Per funzionare correttamente, Clerk richiede una variabile ambientale **CLERK_PUBLISHABLE_KEY** rappresentante la chiave pubblica del proprio profilo associato a Clerk.

L'inserimento di tutto il layout dell'applicativo all'interno del container Clerk permette l'utilizzo di due componenti: <SignedIn/> e <SignedOut/>. Racchiudendo le schermate all'interno di queste componenti è possibile definire quali sono visualizzabili da tutti gli utenti e quali solamente dagli utenti autenticati. Per gli utenti non autenticati viene mostrata un'unica schermata contenente un pulsante di login tramite OAuth 2.0 (mostrato in Figura 3.5), impostato per l'utilizzo di un account Google. Una volta effettuato l'accesso vengono fornite tutte le informazioni sull'utente e la sessione in chiaro. Inoltre, viene fornito anche un token criptato utilizzando la propria chiave privata, contenente informazioni sensibili come l'id dell'utente e della sessione. Questo token, verrà utilizzato in seguito per autenticare le eventuali richieste effettuate verso il backend dell'applicativo.

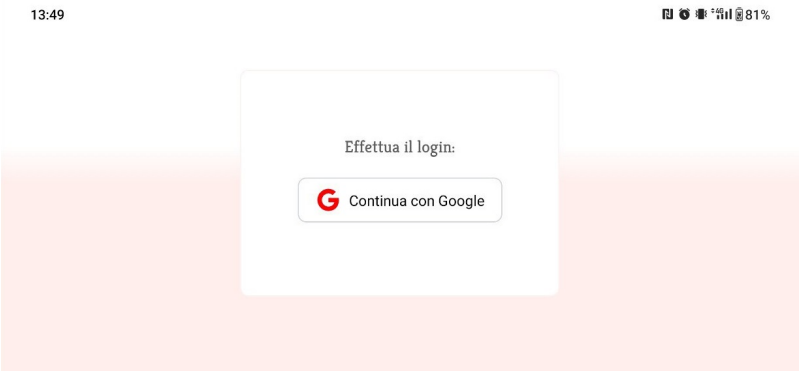

Figura 3.5: Schermata di Login mediante OAuth

Per cercare di ottimizzare la fase di login è stata definita anche una token cache in modo da evitare all'utente di ri-effettuare l'autenticazione ogni volta che accede all'applicazione.

3.2.5 Home Page

La home page dell'applicativo, mostrata in Figura 3.6, è composta da vari PictogramCard che rappresentano le varie sezioni dell'applicativo. Come per tutte le pagine utilizzate dall'utente finale, la navigazione è effettuata mediante pittogrammi per favorire l'accessibilità a persone con difficoltà nell'utilizzo di interfacce standard.

Figura 3.6: Home page di PictoAI

3.2.6 Giochiamo

La Sezione Giochiamo, mostrata in Figura 3.7, permette di accedere ai vari mini giochi educativi. Inizialmente era stato implementato solamente il minigioco "Che cos'è?", in quanto le componentistiche dell'intelligenza artificiale per generare i giochi non erano ancora state ultimate (anche "Che cos'è?" viene generato senza AI attualmente). Inoltre, va sottolineato che tutte i mini-giochi proposti durante la fase di scoping, erano soltanto possibili idee che avrebbero potuto far parte dell'applicativo. Lo scopo di questa tesi infatti non è la loro effettiva implementazione, ma la realizzazione di un "framework" contenente un'interfaccia che in futuro possa essere ampliata ed espansa da una moltitudine di giochi, elaborati in collaborazione con gli esperti del dominio, per incentivare l'utilizzo da parte degli utenti e fornire un buon supporto all'apprendimento.

Una volta selezionato il gioco viene mostrata una schermata che permette di selezionare una categoria tra quelle disponibili sulla quale basare la generazione del gioco. Una categoria può venire pre-impostata dalle impostazioni in modo che sia pre-selezionata di default, permettendo al supervisore di focalizzare l'apprendimento dell'utente su un determinato contesto educativo.

Figura 3.7: Pagina "Giochiamo" di PictoAI

Che cos'è?

"Che cos'è?" è un mini-gioco nel quale l'utente deve individuare l'elemento contenuto nell'immagine al centro dello schermo tra i vari pittogrammi visualizzati, come mostrato in Figura 3.8. Sfruttando le funzioni accessibili da useHuggingFace, descritto in Sezione 3.2.2, viene effettuata una richiesta per generare il gioco, ottenendo in risposta: il testo della domanda, i possibili pittogrammi selezionabili (di cui uno indicato come corretto) e l'immagine da mostrare in formato base64. Una volta caricato il gioco, il testo viene letto tramite TTS.

Figura 3.8: Mini gioco "Che cos'è?", della Sezione "Giochiamo"

Appena l'utente seleziona una risposta, viene caricata la pagina del risultato, riproducendo un certo suono in base al fatto che quest'ultima sia corretta o errata. Dalla schermata dei risultati, mostrata in Figura 3.9, è possibile poi visualizzare la soluzione del quesito, giocare nuovamente o uscire dal gioco.

Figura 3.9: Schermata dei risultati nel gioco "Che cos'è?"

3.2.7 Leggiamo

La Sezione Leggiamo, mostrata in Figura 3.10, permette all'utente di visualizzare un libro formato da pittogrammi. Dalla schermata principale vengono visualizzate le copertine di tutti i libri disponibili, all'interno di un componente Carousel [11]. All'installazione dell'applicativo vengono forniti solamente due libri di default ma, dalla pagina di impostazioni e come descritto in Sezione 3.2.10, è possibile generare altri libri a partire da un file testuale fornito dall'utente.

Figura 3.10: Pagina "Leggiamo" di PictoAI

Una volta selezionato un libro viene mostrata la schermata di lettura, come in Figura 3.11. Da questa schermata è possibile leggere mediante TTS tutti i pittogrammi mostrati, premendo sul pulsante sottostante, che corrisponde alla parola "Leggi".

Durante la lettura, il pittogramma selezionato viene evidenziato di giallo e su di esso appare un indicatore a forma di mano per permettete all'utente di

[11] https://www.npmjs.com/package/react-native-reanimated-carousel

associare la parola ascoltata al pittogramma corrispondente. Il pulsante "Leggi" viene sostituito dalla voce "Pausa" durante la lettura. Inoltre, premendo direttamente su un pittogramma, la lettura ricomincerà a partire da esso, e continuerà fino all'ultimo pittogramma mostrato. Premendo sulle zone a lato dello schermo è possibile invece cambiare pagina.

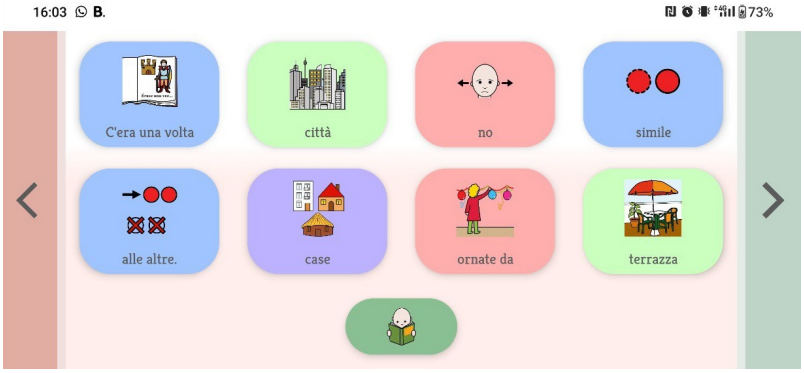

Figura 3.11: Schermata di lettura della pagina "Leggiamo"

Per effettuare la lettura sequenziale di tutti i pittogrammi è stato sfruttato l'hook useEffect, che permette di gestire effetti collaterali in un componente funzionale, eseguendo codice in risposta a cambiamenti nel componente stesso o nelle sue dipendenze. In questo caso, ad ogni sua esecuzione, viene identificato l'indice del pittogramma che è in fase di lettura, estratto il testo e richiamato Companion Store 3.2.3, usufruendo di una funzione di callback che incrementa l'indice del pittogramma letto quando la lettura di quello corrente è terminata.

```
1  useEffect(() => {
2    (async () => {
3      // Checks if the index is defined and if it's the last one
4      const currentID = selectedPictograms[readIndex];
5      const currentPictogram = currentID
6      ...
7      const currentText = currentPictogram
8      ...
9      if (currentText)
10       await companionStore.speak(
11         currentText,
12         ...
13         async () => {
14           if (readingOne) await resetSpeech();
15           else setReadIndex(readIndex + 1);
16         },
17       );
18   })();
19 }, [readIndex, selectedPictograms]);
```

La funzione viene poi rieseguita ogni volta che l'indice del pittogramma letto cambia o quando viene selezionato un pittogramma differente.

3.2.8 Parliamo

La Sezione Parliamo, mostrata in Figura 3.12, semplifica all'utente la comunicazione con le persone nelle immediate vicinanze. Accedendo alla pagina vengono mostrati una serie di pittogrammi, selezionati da esperti del dominio, ritenuti basilari o comunque più frequentemente richiesti. La schermata di selezione dei pittogrammi è scorrevole e, nel caso dei pittogrammi predefiniti, il loro numero corrisponde a sedici, ma potrebbe variare in seguito ad ulteriori incontri con gli esperti. Nel caso in cui un utente prema su un pittogramma, questo viene letto da TTS e aggiunto alla frase attualmente composta, mostrata nella parte alta dello schermo. Quindi, i pittogrammi selezionabili vengono scambiati con quelli previsti dall'IA, mostrandoli in ordine di probabilità di essere scelti come successivi. Durante il caricamento dei pittogrammi successivi viene mostrato un componente Spinner [12].

Dato che la quantità dei pittogrammi non ha un limite, è implementata anche una loro suddivisione in categorie, per facilitare all'utente la ricerca. Tutte le categorie visualizzate (fino ad un massimo di 6) sono selezionabili dalla pagina di Impostazioni 3.2.10. Una volta selezionata una categoria, i pittogrammi mostrati vengono aggiornati, effettuando una nuova chiamata verso la componente di IA per indicare la categoria selezionata. Nel caso in cui un pittogramma venga invece selezionato con una categoria già impostata, quest'ultima viene deselezionata per permettere all'utente di continuare una frase di senso compiuto con i nuovi pittogrammi predetti dall'IA (es. se un utente seleziona un pittogramma corrispondente ad un frutto è improbabile che voglia continuare a selezionare frutti).

Nel caso in cui l'utente abbia selezionato la modalità pittogrammi ingranditi dalle Impostazioni, viene mostrata una sola riga di pittogrammi, con un'altezza maggiore e viene ingrandita anche la barra di selezione delle categorie.

Premendo su uno dei pittogrammi selezionati, questo viene poi rimosso. Sulla pagina sono presenti anche tre pulsanti:

- Leggi: pittogramma collocato sulla destra della pagina che se premuto effettua la lettura mediante TTS di tutti i pittogrammi attualmente selezionati, evidenziando quello attualmente letto

- Cancella: pittogramma collocato sulla destra della pagina che elimina tutti quelli attualmente selezionati

[12] https://reactnative.dev/docs/activityindicator

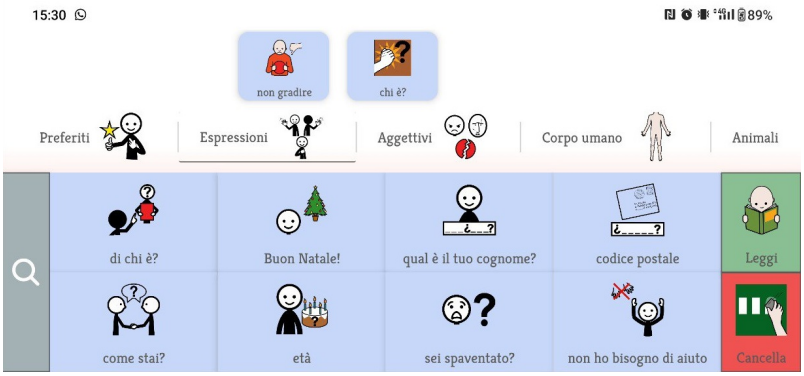

Figura 3.12: Pagina "Parliamo" di PictoAI

- Cerca: icona collocata sulla sinistra della pagina. In questo caso non è stato utilizzato un pittogramma in quanto la funzionalità di ricerca dovrebbe essere utilizzata solamente da un tutor o da un'altra persona che assiste l'utente quando non riesce a trovare un determinato pittogramma. Premendo su di questo viene poi mostrato una modale per la ricerca dei pittogrammi, descritto in Sezione 3.2.1

La pagina Parliamo può anche essere usata per chiedere all'utente in input delle frasi, sotto forma di pittogrammi, da utilizzare in altre pagine come *Diario*. Per utilizzarla a tale scopo è necessario impostare il parametro "inputID" quando viene effettuato il routing verso la pagina, cosicché possa essere utilizzato per identificare la tipologia di richiesta da InputStore, descritto in Sezione 3.2.3. Una volta identificata la richiesta vengono caricati eventuali pittogrammi forniti in input e il pittogramma "Leggi" viene sostituito da "Conferma" che, se premuto, riporta alla pagina che ha effettuato la richiesta.

3.2.9 Diario

La Sezione Diario, mostrata in Figura 3.13, permette all'utente di scrivere in autonomia delle serie di frasi che si suppone vengano fatte ascoltare, in seguito, ad un tutore.

Appena aperta la pagina, in automatico viene selezionata la data odierna, che sarà possibile cambiare utilizzando le frecce sulla parte alta dello schermo. Nella pagina sarà sempre presente, in fondo, un pulsante col pittogramma corrispondente ad aggiungi che, se premuto, permetterà di inserire una frase generata dalla pagina *Parliamo*. Il diario è formato da una lista di frasi scorrevole in verticale, per permettere di inserire una quantità indefinita di frasi. Ciascuna frase, una volta inserita, può essere letta utilizzando il pulsante "Leggi" collocato alla sua sinistra, oppure modificata mediante il pulsante

"Modifica" che reindirizza alla pagina *Parliamo*, impostando in input la frase corrente. Premendo su un pittogramma questo verrà poi letto. Tutte le pagine di diario contenenti almeno un elemento, vengono memorizzate in automatico sul Database in seguito alla loro creazione e aggiornate in seguito a ciascuna modifica. Nel caso in cui nelle frasi sia presente un pittogramma personalizzato, quando questo viene rimosso verrà eliminato anche da tutte le pagine di diario.

Figura 3.13: Pagina "Diario" di PictoAI

Le date di ciascuna pagina di Diario vengono memorizzate in formato ISO.

3.2.10 Impostazioni

La Sezione Impostazioni, mostrata in Figura 3.14, è accessibile da una delle icone presenti a fondo pagina. All'interno di questa pagina, destinata solamente al tutore dell'utente, è possibile: modificare pittogrammi, scegliere quelli preferiti, modificare le preferenze dei pittogrammi, aggiungere o eliminare libri e selezionare le categorie visualizzate.

Figura 3.14: Pagina "Impostazioni" di PictoAI

Preferiti

Nella pagina dei preferiti, mostrata in Figura 3.15, è possibile selezionare dei pittogrammi che verranno poi mostrati nella relativa Sezione presente all'interno della pagina Parliamo. Premendo sul pulsante aggiungi viene aperta una modale per la ricerca dei pittogrammi, descritto in Sezione 3.2.1, che mostrerà inizialmente i pittogrammi personalizzati permettendo di aggiungerli rapidamente ai preferiti.

Al di fuori della modale, premendo su uno dei pittogrammi preferiti, quest'ultimo verrà rimosso.

Figura 3.15: Gestione dei pittogrammi Preferiti dalla pagina "Impostazioni"

Pittogrammi

Dalla pagina delle impostazioni dei pittogrammi, questi possono essere personalizzati a proprio piacimento dall'utente, aggiungendone di nuovi mediante un pulsante che apre un PictogramCustomizationModal, descritto in 3.2.1, o rimuovendo quelli esistenti, mediante una modale simile a quella di ricerca dei pittogrammi. Nella pagina sono presenti anche alcune checkbox:

- Mostra aggettivi: permette di impostare la presenza di aggettivi tra i pittogrammi e attivandola, ogni volta che viene fatta una richiesta verso l'IA, verrà inclusa anche questa specifica impostazione.

- Mostra colori in base alla categoria: permette, se attivata, di mostrare i pittogrammi con uno sfondo associato alla categoria

- Pittogrammi ingranditi: permette di visualizzare, in alcune pagine, pittogrammi dalle dimensioni incrementate

Libri

Dalla pagina delle impostazioni dei libri è possibile aggiungere o rimuovere libri a proprio piacimento per la pagina "Leggiamo". La schermata per aggiungere un libro è una modale molto simile a PictogramCustomizationModal, descritta in 3.2.1, ma a differenza di quest'ultima, premendo sul pittogramma, viene aperta la selezione dell'immagine di copertina, oltre all'aggiunta di un pulsante "Seleziona un libro da importare" che permette di scegliere un file testuale da convertire in pittogrammi.

Categorie

La pagina di impostazioni delle categorie, mostrata in Figura 3.16, permette di selezionare le categorie da visualizzare ogni qualvolta vi sia la necessità di selezionarne una. Per scegliere le categorie basterà premere sul pulsante avente come testo la categoria desiderata, che nel caso in cui fosse già presente tra quelle visualizzate viene rimossa, altrimenti aggiunta.

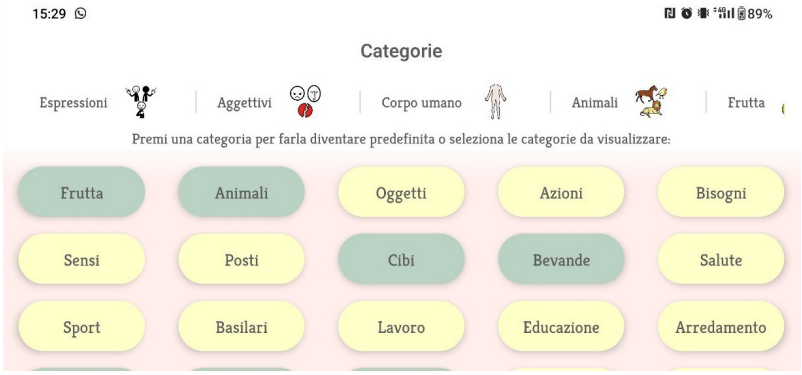

Figura 3.16: Gestione delle Categorie dalla pagina "Impostazioni"

3.3 Backend

Come descritto nel Capitolo 2, come Backend è stato utilizzato Next.js, impostando come unica route HTML una pagina 404 in modo da esporre unicamente gli endpoint per le API. Queste ultime sono implementate mediante i file .tsx contenuti nella cartella "src/pages/api" e utilizzate per interfacciare il client con il database attraverso Prisma, operazione descritta nella Sezione successiva. Di seguito è riportata una descrizione delle varie API e del middleware utilizzato per effettuare l'autenticazione delle richieste.

3.3.1 API

/api/customPictogram

Tipo richiesta: POST
Descrizione: Aggiunge un pittogramma personalizzato con i dati forniti.
Dati in input:

- oldId?: ID del pittogramma originale associato
- text?: testo del nuovo pittogramma
- image?: immagine in formato base64 del nuovo pittogramma

Output: Oggetto con campo "pictogram" contenente il nuovo pittogramma creato, o un campo "error".

/api/diary

Tipo richiesta: POST
Descrizione: Crea una pagina del diario con i dati forniti. Se è già presente una pagina con la stessa identica data, questa viene poi aggiornata con i nuovi pittogrammi presenti all'interno della richiesta
Dati in input:

- date: data della nuova pagina di diario, in formato ISO [13]
- pictograms: array contenente gli id dei pittogrammi presenti in pagina

Output: Oggetto con campo "page" contenente la nuova pagina creata, o un campo "error"

/api/favourite

Tipo richiesta: POST
Descrizione: Aggiorna il profilo dell'utente con i nuovi elementi preferiti
Dati in input:

- favourites: array contenente gli id dei pittogrammi preferiti

Output: Oggetto con campo "user" contenente le informazioni del profilo utente o un campo "error"

[13] https://it.wikipedia.org/wiki/ISO_8601

/api/user

Tipo richiesta: POST
Descrizione: Restituisce l'utente corrispondente all'ID Clerk fornito, che nel caso in cui non fosse presente, viene creato con i dati ricevuti in input.
Dati in input:

- email?: email del nuovo utente

Output: Oggetto con campo "user" contenente il profilo dell'utente o un campo "error"

3.3.2 Middleware

Il middleware è stato configurato per autenticare ed ottenere le informazioni sul richiedente ad ogni chiamata effettuata attraverso le API messe a disposizione. Per la sua implementazione, è bastato creare un file middleware all'interno della cartella "src", che viene riconosciuto in automatico da Next.js. Per un corretto funzionamento delle API e dell'autenticazione è necessario impostare due variabile d'ambiente:

- DATABASE_URL: stringa contenente l'URL del database Planetscale
- CLERK_JWT_URL: stringa contenente la chiave pubblica, fornita da Clerk, per autenticare i JWT (JSON Web Token)

Una volta impostate queste variabili, ciascuna richiesta verrà autenticata estraendo il token dall'header "Authentication" e decriptando quest'ultimo mediante la corrispondente chiave pubblica e la libreria Jose [14]. Ottenuto il valore del token, sarà poi possibile visualizzare anche delle informazioni utili a verificare la validità della sessione, negando la richiesta nel caso in cui questa sia scaduta, visualizzare l'id dell'utente che viene impostato come cookie prima di inoltrare la richiesta effettiva alle API. In seguito il codice dell'autenticazione:

```
async function authenticateRequest(token: string) {
  try {
    const JWKS = createRemoteJWKSet(new URL(process.env.CLERK_JWT_URL!));

    // Verify the given token
    const { payload } = await jwtVerify(token.replace("Bearer ", ""), JWKS);
    const response = NextResponse.next();

    // Set a cookie to pass user ID
    response.cookies.set("user", payload.sub ? payload.sub : "");
    return response;
  } catch (e) {
```

[14] https://www.npmjs.com/package/jose

```
13      console.log(e);
14      return NextResponse.json(
15        {
16          message: "Authentication failed: Token could not be verified",
17        },
18        { status: 401, headers: { "content-type": "application/json" } },
19      );
20    }
21  }
```

3.4 Database

Come piattaforma database serverless è stato utilizzato Planetscale, già descritto in Sezione 2.2.2, che permette di usufruire di un database gratuito con limiti estremamente generosi (5GB di spazio, 1 miliardo di letture/mese, 10 milioni di scritture/mese). La creazione di un database dall'interfaccia web è triviale: è necessario registrarsi, premere sul pulsante aggiungi database e scegliere il nome e stato in cui verrà ospitato quest'ultimo. Una volta creato, verrà fornita una stringa di connessione, una per ogni profilo e con determinati permessi, per effettuare la connessione attraverso prisma e definire la struttura delle tabelle. Il diagramma ER del database è mostrato in Figura 3.17.

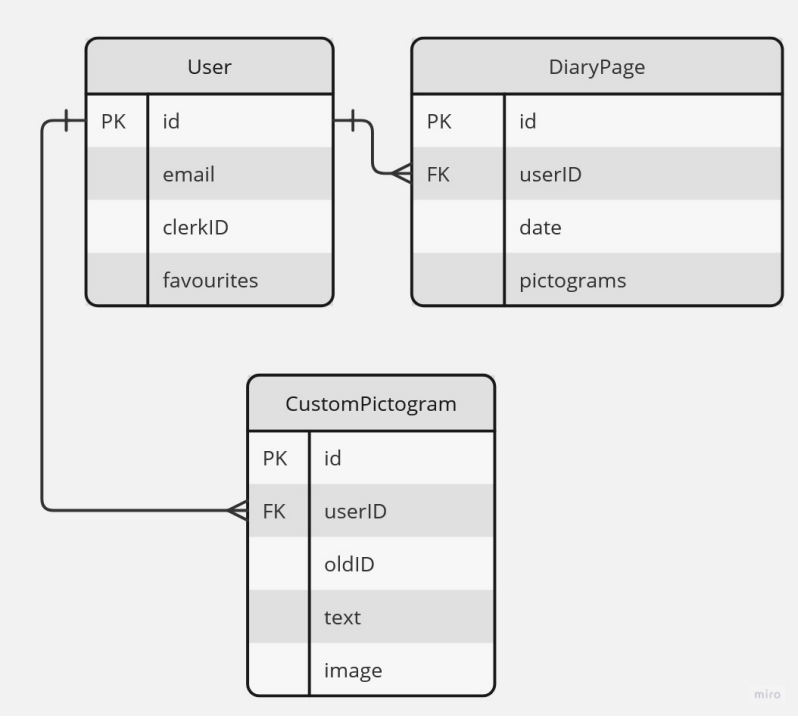

Figura 3.17: Schema ER rappresentante la struttura del Database su Planetscale

In seguito una descrizione di tutte le entità e dei relativi attributi:

- User: utente registrato nell'applicativo
  - id (Chiave primaria, default(cuid)): id dell'utente
  - email (unique): email dell'utente
  - clerkID (unique): id restituito da Clerk dell'utente
  - favourites (opzionale): stringa che rappresenta un array contenente gli id dei pittogrammi preferiti
- DiaryPage: pagina di diario di un utente
  - id (Chiave primaria, default(cuid)): id della pagina di diario
  - userID (Chiave esterna): id dell'utente che ha generato la pagina
  - date (VarChar[32], unique): stringa che che rappresenta la data della pagina in formato ISO
  - pictograms: stringa che rappresenta una matrice contenente gli id dei pittogrammi contenuti nella pagina, suddivisi in entry
- CustomPictogram: pittogramma personalizzato da un utente
  - id (Chiave primaria, default(cuid)): id del pittogramma personalizzato
  - userID (Chiave esterna): id dell'utente che ha personalizzato il pittogramma
  - oldID (opzionale): id del pittogramma originale associato
  - text (opzionale, VarChar[128]): testo del pittogramma personalizzato
  - image (opzionale, MediumText): immagine in formato base64 del pittogramma personalizzato

3.4.1 Prisma

Per la generazione dello schema è stato utilizzato Prisma, definendo nella stringa di connessione un profilo con privilegi di modifica ed eseguendo il seguente comando:

```
1    pnpm with-env prisma db push
```

In automatico verrà generato lo schema dal file schema.prisma per poi essere caricato sul database.

3.5 Deployment

La fase di deployment è suddivisa in due parti, la build dell'applicativo mobile e il deployment del backend, descritti nelle successive Sezioni.

3.5.1 Build APK

Lo scopo di questa tesi prevede che l'applicativo mobile venga rilasciato solamente sotto forma di APK e distribuito mediante Github. Se in futuro si desiderasse rilasciare PictoAI anche sui relativi store delle varie piattaforme (Android, iOS), il processo non richiederà alcuna particolare complicazione e il tutto viene descritto nella documentazione ufficiale di expo[15].

Per ottenere l'APK, seguendo la documentazione ufficiale [16], è necessario impostare correttamente le variabili d'ambiente, definendo per ciascuna un campo nella Sezione extra del file app.config.json, che verrà poi utilizzato nel codice mediante Constants.expoConfig?.extra?, necessitando una modifica del campo eas.projectId. In seguito è necessario poi definire i seguenti campi nel file eas.json:

- developmentClient: true
- android.buildType: "apk"

Infine, per avviare il processo di build basterà utilizzare il seguente comando (sostituendo NOME_BUILD col nome della build nel file):

```
1    eas build -p android --profile "NOME_BUILD"
```

Una volta avviato si verrà messi in una coda (durata ad esempio 23m nell'ultima build), in seguito alla quale il file .apk sarà generato automaticamente e reso disponibile dal proprio profilo expo, come mostrato in Figura **??**. Dalla seguente pagina sarà poi possibile scaricare il file apk o installarlo direttamente su un dispositivo mobile tramite codice QR.

3.5.2 Hosting

Il deployment del backend è effettuato da Vercel, descritto in Sezione 2.2.2. Servizio per hosting di siti web e serverless function, offre un servizio gratuito dai limiti particolarmente generosi: nel caso di un applicativo Next.js deployato le limitazioni comprendono un massimo di dodici funzioni serverless con tempo massimo di esecuzione pari a 10 secondi.

[15] https://docs.expo.dev/submit/introduction/
[16] https://docs.expo.dev/build-reference/apk/

Per effettuare il deployment effettivo è bastato effettuare il login sul sito di Vercel, importare la repository github del progetto [17] e aggiungere le variabili d'ambiente. Una volta effettuati i passaggi precedenti Vercel effettua il deployment dell'applicativo in automatico e ad ogni commit, fornendo un URL pubblico per l'accesso al servizio e un pannello di gestione del dominio come mostrato in Figura 3.18.

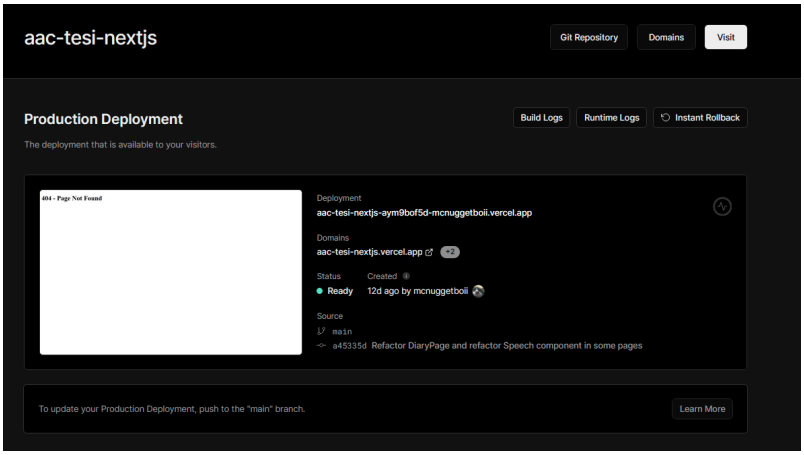

Figura 3.18: Pannello di gestione del dominio su Vercel

[17] https://github.com/mcnuggetboii/AAC-Tesi

Capitolo 4

Validation

Per la validazione del progetto è stato svolto un ultimo incontro con gli esperti menzionati in Sezione 2.1.1. Il focus dell'incontro è stato posto sulla raccolta di un feedback finale, sulla verifica dell'adempimento ai requisiti di accessibilità e usabilità di PictoAI e sull'adempimento alle Conditions of Satisfaction definite in 2.1.4.

Durante l'incontro sono state fatti alcuni appunti riguardanti l'interfaccia di PictoAI come:

- Implementare la possibilità di scegliere se utilizzare i colori o meno in base alla categoria del pittogramma
- Nella pagina di lettura il testo va posto sopra all'immagine
- Possibilità di personalizzare il colore di un pittogramma
- Rifare la componente per la selezione delle categorie, eliminando le icone in quanto non interpretabili dagli utenti finali
- Aggiungere un'Agenda con le attività settimanali da svolgere

A parte l'ultimo appunto, che è stato segnato come possibile sviluppo futuro, tutti gli altri sono stati risolti, confermando la validità delle soluzioni.

Oltre agli appunti elencati, gli esperti si sono mostrati molto soddisfatti della soluzione proposta giudicandola più che appropriata, elogiando le funzionalità proposte da ciascuna Sezione come un "aiuto molto grande e apprezzato, sia per i bambini che per gli educatori". Infine, tutte le Conditions of Satisfaction sono state validate e accertate del loro soddisfacimento.

Capitolo 5

Verbalizzazione Knowledge Graph

Come progetto complementare allo sviluppo dell'applicativo, è stato realizzato un ulteriore lavoro per la verbalizzazione di Knowledge Graph utilizzando modelli di IA. Il lavoro ha come obiettivo la generazione di frasi da utilizzare nella componente *Giochiamo*: il modello, una volta addestrato a generare frasi semplici e di senso compiuto, viene impiegato nella generazione di alcuni giochi. Un possibile esempio è il gioco "Quiz", descritto in Sezione 2.1.3, nel quale viene generata una frase in cui manca un elemento tra soggetto, predicato o oggetto e l'utente deve indovinare l'elemento mancante; in questo caso la frase verrebbe generata dal modello, fornendo in input un subset contenente l'elemento che si desidera far indovinare all'utente per poi rimuoverlo dalla frase generata. Un altro possibile impiego deriverebbe dalla generazione di frasi abbastanza semplici da poter essere utilizzate per addestrare un altro modello, attualmente in sviluppo dal Dott. Frisoni, per la conversione di testo in pittogrammi. Questo per compensare alla mancanza di materiale, disponibile pubblicamente, contenente frasi facilmente comprensibili anche da persone con problemi comunicativi.

Lo scopo del progetto è la generazione di frasi semplici, comprensibili anche da persone con particolari difficoltà o disturbi dello spettro autistico, mediante un modello basato su transformers [33]. Nello specifico, si punta a ottenere dei set di concetti (o parole), coerenti tra di loro, da utilizzare per la generazione di frasi. Questo specifico task è stato trattato e descritto nel paper CommonGen [34], che ha anche elaborato e proposto una versione fine tuned del modello T5 [35], addestrata ad utilizzare set di parole per la generazione di frasi. Oltre alla generazione dei vari set di concetti si ad adattare il modello proposto da CommonGen, effettuando il fine tuning per ottenere frasi più semplici e adatte al contesto di questa tesi.

5.1 ConceptNet

Vista la necessità di ottenere set di parole con concetti correlati tra di loro si è pensato di utilizzare un Knowledge Graph: un grafo composto da nodi, rappresentanti risorse o entità, e le varie connessioni che mostrano le relazioni tra i nodi. Sul Web sono disponibili e pubblici innumerevoli grafi di questo tipo che permettono di ottenere informazioni sotto forma di fatti (due nodi connessi da una relazione), sugli argomenti più vari. In particolare per questo progetto è risultata interessante ConceptNet [36], Knowledge Graph che connette parole o frasi di varie lingue mediante relazioni significative. Il funzionamento di quest'ultimo può essere riassunto dallo schema in Figura 5.1.

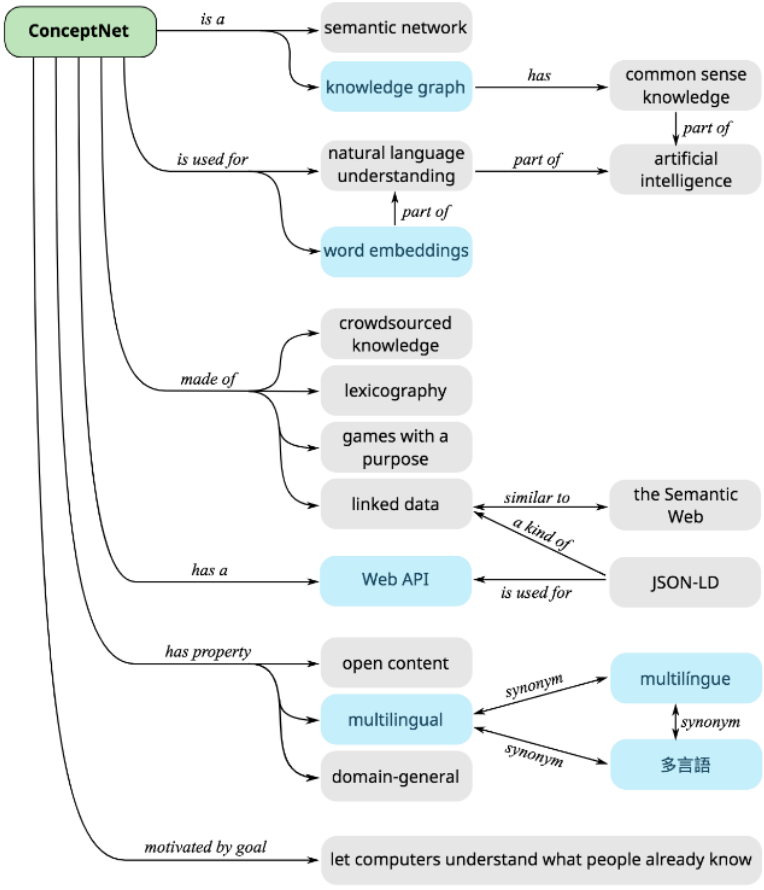

Figura 5.1: Schema rappresentativo di ConceptNet
Preso da ConceptNet

Tutti i nodi e le connessioni presenti su ConceptNet sono resi disponibili mediante delle API o da un unico file scaricabile in formato ".csv.gz" [1]. Partendo

[1]https://s3.amazonaws.com/conceptnet/downloads/2019/edges/conceptnet-assertions-

da questo file sono state elaborate alcune possibili soluzioni, descritte nelle Sezione seguenti, per estrarre i set di parole da utilizzare nella generazione delle frasi.

5.2 Approccio Naive

Il primo approccio che si è cercato di utilizzare è un approccio naive, o ingenuo, e consiste nella generazione di un dizionario di concetti correlati, studiando e selezionando manualmente le relazioni più significative presenti tra i nodi [2]. Una volta studiate le relazioni, i concetti sono stati raggruppati in delle categorie ritenute utili nel contesto di questa tesi, come per esempio frutta, cibo o animali.

Per individuare tutti gli elementi appartenenti ad una categoria, sono stati trovati tutti i nodi con una relazione avente come testa il nome della categoria. Per esempio esempio partendo dal concetto "food", sono stati ottenuti nodi come "banana" o "bbq", tramite relazione "isA", la quale indica che il secondo nodo della relazione fa parte della categoria rappresentata dal primo. Quindi, per ciascuna categoria, sono state selezionate alcune relazioni ricorrenti e significative come ad esempio, per un elemento appartenente alla categoria cibo, la relazione "ReceivesAction", che indica un'azione che può essere applicata al cibo, come "fritto" o "bollito".

Infine, sono stati generati i set di parole, selezionando casualmente un concetto appartenente ad una categoria e ad alcuni concetti associati, mediante una relazione considerata significativa, scelti anch'essi casualmente. Da tale approccio si è potuto osservare un ottimo livello di coerenza tra le parole; il problema più grande è però dato dalla quantità limitata di possibili combinazioni che è possibile generare. Difatti, le relazioni che sono state scelte, seppur significative, non erano sempre presenti nei concetti correlati ad una categoria. Inoltre, si è anche notato come molti concetti correlati ad una categoria non siano connessi a quest'ultima mediante una relazione "isA, limitando perciò il numero di combinazioni in output.

5.3 Approccio KELM

KELM (Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training) [37] è un approccio proposto durante la *Conference of the North American Chapter of the Association for Computational Linguistics* per la verbalizzazione dei Knowledge Graph. In questo

5.7.0.csv.gz

[2]https://github.com/commonsense/conceptnet5/wiki/Relations

specifico paper viene descritto il processo che è stato utilizzato per verbalizzare l'intero Knowledge Graph Inglese di Wikidata. L'approccio proposto punta ad allineare, per ogni entità, le frasi presenti nella relativa pagina di Wikipedia con tutte le triple, dell'intero grafo, aventi l'entità come soggetto, individuando raggruppamenti di triple correlate in base al conteggio di co-occorrenze delle relazioni, calcolato dall'algoritmo mostrato in Figura 5.2.

```
all_triple_sets ← {}
rel_pairs ← {}
depth ← 5
for all r_i ∈ KG do
    P ← {(r_j, c_ij) ∀(r_i, r_j, c_ij) ∈ train_align_counts}
    rel_pairs(r_i) ← maxheap(P)
end for
for all entities s ∈ KG do
    R ← {(r, o) ∀(s, r, o) ∈ KG}
    while R ≠ ∅ do
        triple_set ← {}
        (r_1, o_1) ← random(R)
        triple_set.add(s, r_1, o_1)
        R.remove(s, r_1, o_1)
        KG.remove(s, r_1, o_1)
        for i = 2 to depth do
            r_i ← NONE
            M ← rel_pairs(r_{i-1})
            while M ≠ ∅ do
                (r_j, c_ij) ← M.next
                if r_j ∈ R then
                    r_i ← r_j
                    (r_i, o_i) ← R.get(r_i)
                    triple_set.add(s, r_i, o_i)
                    R.remove(s, r_i, o_i)
                    KG.remove(s, r_i, o_i)
                    break
                end if
            end while
        end for
        all_triple_sets.add(triple_set)
    end while
end for
```

Figura 5.2: Algoritmo usato da KELM per raggruppare le triple in base alle occorrenze
Preso dal [37]

L'approccio utilizzato per effettuare il raggruppamento è stato ritenuto interessante ai fini del progetto e quindi replicato. Purtroppo, dai risultati ottenuti si è notato un basso livello di coerenza tra le parole ottenute nei raggruppamenti, dovuto alla quantità molto più bassa (trentaquattro) di relazioni presenti in ConceptNet rispetto a Wikidata.

5.4 Approccio Knowledge Graph Embedding

Durante la ricerca di una possibile soluzione, sono stati trovati alcuni articoli [38] e guide [3] che tentavano di ottenere raggruppamenti tra concetti ottenuti da Knowlege Graph mediante il loro embedding seguito dall'applicazione di un algoritmo di clustering. Gli approcci adottati sono stati ritenuti in linea con lo scopo del progetto pertanto si è cercato di replicarli.

Per l'embedding dei dati sono stati utilizzati due algoritmi:

- TransE [39]: modello progettato per effettuare l'embedding di entità e relazioni in modo da preservare le loro relazioni semantiche. In particolare, questo modello punta a rappresentare ogni tripla (testa, relazione, coda) in modo tale che la relazione tra la testa e la coda possa essere approssimata sommando la rappresentazione della relazione alla rappresentazione della testa.

- DistMult [40]: modello di Knowledge Graph embedding che utilizza multiple matrici per rappresentare le relazioni tra le entità. Abbastanza simile a RESCAL, ma anziché utilizzare matrici complesse, riduce il numero di parametri delle relazioni utilizzando solo una matrice diagonale. Dal punto di vista computazionale, DistMult è simile a TransE, ma utilizza un'interazione moltiplicativa, mentre TransE utilizza un'interazione additiva.

Entrambi gli embedding sono stati effettuati su un sample corrispondente al 12% dei dati, per delle limitazioni computazionali dell'hardware utilizzato, suddividendoli in subset di training (80%), testing(10%) e validation(10%), mediante la libreria pykeen [4]. Nel caso di TransE il modello è stato eseguito per 20 epochs mentre per DistMult 50 epochs.

Per effettuare il clustering degli embedding ottenuti, sono stati scelti tre algoritmi di clustering:

- HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [5]: esegue DBSCAN su diversi valori di epsilon e integra

[3] https://docs.ampligraph.org/en/2.0.0/tutorials/ClusteringAndClassificationWithEmbeddings.html
[4] https://github.com/pykeen/pykeen
[5] https://github.com/scikit-learn-contrib/hdbscan

il risultato per trovare un clustering che offre la migliore stabilità su epsilon, individuando, a differenza di DBSCAN, cluster con densità variabili. È stato selezionato in quanto non richiede di definire il numero effettivo di cluster.

- OPTICS (Ordering Points To Identify the Clustering Structure): algoritmo basato su DBSCAN, permette come HDBSCAN di individuare cluster significativi in dati con densità variabile e non richiede di definire in anticipo il numero di cluster.

- MiniBatchKMeans: algoritmo basato su K-means che lavora su mini batch casuali dei dati anziché sull'intero set, rendendo l'algoritmo più veloce e adatto per l'elaborazione di grandi quantità di dati. È stato selezionato per verificare il risultato ottenuto definendo in anticipo il numero di cluster.

Infine, su entrambe le rappresentazioni, TransE e DistMult, sono stati applicati tutti gli algoritmi di clustering. Nel caso di HDBSCAN, si è osservato che rispettivamente il 56%, per TransE, e il 63%, per DistMult, dei concetti non sono stati assegnati ad un cluster, ma catalogati come rumore. Sono quindi stati campionati alcuni cluster ed è stata analizzata la correlazione tra le entità ottenute. In tutti i cluster studiati è stata notata una bassa correlazione tra i concetti, pertanto non sono stati considerati utili ai fini del progetto. Un esempio di cluster così ottenuto è il seguente:

```
['/c/en/beef', '/c/en/brisket', '/c/en/convivial/a', '/c/en/gastrosophy/n',
↪  '/c/en/gerontophagy/n', '/c/en/glutton', '/c/en/manducation/n',
↪  '/c/en/necrophagy/n', '/c/en/phagia', '/c/en/yolk']
```

Nel caso di OPTICS, i risultati sono stati ritenuti non significativi, in quanto il 99% dei concetti è stato catalogato come rumore.

Come ultimi, sono stati analizzati i cluster ottenuti da MiniBatchKMeans, ma anche in questo caso tutti i cluster non sono stati ritenuti adeguati all'utilizzo per la generazione di frasi. Un esempio di cluster ottenuto è il seguente:

```
['/c/en/aeradio/n',
  '/c/en/bardiglio/n',
  '/c/en/beef',
  '/c/en/byground/n',
  '/c/en/condominium/n',
  '/c/en/earthed/a',
  '/c/en/fishmeal/n',
  '/c/en/geoglyph/n',
  '/c/en/goalsquare/n',
  '/c/en/groundworker/n',
  '/c/en/highlining/n',
  '/c/en/implantation/n',
```

```
  '/c/en/inground/a',
  '/c/en/mafe/n',
  '/c/en/oviparity/n',
  '/c/en/palmigrade/a',
  '/c/en/reground/v',
  '/c/en/réseau/n',
  '/c/en/sfiha/n',
  '/c/en/warlott/n']
```

5.5 Fine Tuning

Una volta elaborate alcune metodologie per l'estrazione dei concetti, il focus è stato posto sul fine tuning del modello proposto da CommonGen [6], per rendere le frasi generate più adatte al contesto. Per effettuare questo tipo di addestramento, sono stati utilizzati cento esempi di frasi con annessi i set di parole utilizzati per la loro generazione. Questi esempi sono stati scritti manualmente, per garantirne la comprensibilità a persone con bisogni speciali.

Per il fine tuning, i nuovi input sono stati suddivisi in training (70), testing (15) e validation (15). Il modello è stato addestrato con i seguenti parametri:

```
batch_size = 4
args = Seq2SeqTrainingArguments(
    "aac_commongen_t5_finetuned",
    evaluation_strategy = "epoch",
    learning_rate=1e-4,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    save_strategy="steps",
    save_steps=20,
    logging_steps=20,
    save_total_limit=2,
    weight_decay=0.01,
    num_train_epochs=5,
    predict_with_generate=True,
    fp16=True,
)
```

5.6 Risultati

Per la validazione del modello è stata utilizzata la metrica ROGUE [41], applicata al dataset di test. I risultati ottenuti sono i seguenti:

```
{'rouge1': 61.7122,
 'rouge2': 32.0271,
 'rougeL': 56.3103,
 'rougeLsum': 57.2686,
```

[6] https://huggingface.co/mrm8488/t5-base-finetuned-common_gen

```
5  'gen_len': 9.9333}
```

Questi valori, se confrontati con le metriche esposte da CommonGen, sono risultati superiori. Va però considerato che il numero di frasi utilizzate per la validazione è molto limitato. Inoltre, la metrica utilizzata calcola il numero di N-grammi comuni tra la frase generata e quella fornita come punto di riferimento, che nel caso di frasi più corte e semplici è più facile da ottenere.

Infine, il modello è stato utilizzato per generare alcune frasi che sono state confrontate con quelle generate dal modello senza fine tuning. Per la scelta dei set di parole è stato utilizzato l'approccio naive e i risultati ottenuti sono esposti in Tabella 5.1.

Input Set	CommonGen	CommonGen fine-tuned
albatross animal bird	an albatross is a bird that is a small animal	An albatross is an animal that eats birds
red warnings	warnings in red on the wall	Red warnings are displayed on the wall
turkey country ankara	country is a city and a city in the north of turkey	Turkey is a country that has a large population
potato patch	A patch of potatoes is in the middle of a field.	Potatoes are growing in a patch
red apple refrigerator	A refrigerator with a red and white refrigerator with apples	Red apples are in the refrigerator
yellow green	A man is a man in green and yellow.	Yellow and green are the colors of the year
brown eyes	A man with brown eyes is squinting his eyes.	Brown eyes with brown hair
yellow orchids homes	a home with yellow and white orchids	Yellow orchids grow in the homes

Tabella 5.1: Confronto tra le frasi generate dal modello CommonGen base e fine-tuned

Comparando le frasi generate dai due modelli si può già notare che quelle generate dal modello fine-tuned siano più semplici e dirette. Per valutare i risultati in maniera più formale, sono stati generati ulteriori set di esempi col nuovo modello, valutando i risultati mediante una scala di Likert secondo i seguenti criteri: Fluidità, Coerenza e Significatività.

Nella valutazione di una frase, i punteggi per il criterio di **Fluidità** sono stati assegnati come segue:

1. Le frasi contengono numerosi errori grammaticali, una struttura povera e un linguaggio incoerente. È difficile comprenderle e non hanno senso nel contesto fornito.

2. Le frasi presentano alcuni errori grammaticali e incongruenze, richiedendo revisioni significative per renderle più fluide.

3. Le frasi contengono errori o incongruenze minori. Sono in generale comprensibili, ma è possibile migliorarle.

4. Le frasi sono ben strutturate, con solo alcuni errori o incongruenze minori, trasmettendo efficacemente i messaggi previsti, ma potrebbero beneficiare di piccoli miglioramenti.

5. Le frasi sono prive di errori o incongruenze e trasmettono efficacemente il messaggio previsto con chiarezza e precisione.

Coerenza:

1. Le frasi sono altamente incoerenti, mancano di un flusso logico e non riescono a trasmettere messaggi chiari.

2. Le frasi presentano problemi evidenti di coerenza, con difficoltà nel collegare idee e mantenere una progressione logica.

3. Le frasi hanno una coerenza media, con occasionali problemi minori nel collegare idee o mantenere un flusso logico.

4. Le frasi sono per lo più coerenti, collegando efficacemente idee e mantenendo una progressione logica, anche se potrebbero esserci margini per piccoli miglioramenti.

5. Le frasi sono altamente coerenti, con un flusso logico chiaro di idee. Collegano efficacemente i concetti e mantengono una forte progressione.

Significatività:

1. Le frasi mancano di significato, possono essere irrilevanti, prive di senso o non riescono a trattare il contesto fornito.

2. Le frasi presentano problemi evidenti nel trasmettere il significato, possono esserci ambiguità, inesattezze o affermazioni non chiare.

3. Le frasi trasmettono adeguatamente il messaggio previsto, tuttavia potrebbe esserci spazio per migliorare chiarezza e precisione.

4. Le frasi sono per lo più significative, catturando l'essenza del contesto fornito, anche se potrebbero essere possibili leggere migliorie.

5. Le frasi trasmettono accuratamente i messaggi previsti con chiarezza e precisione, offrendo una comprensione chiara del significato del testo.

In seguito i risultati finali:

- Fluency: **4**
- Coherence: **3**

- Meaningfulness: **3**

Questi risultati sono relativi alle frasi generate dal modello fine-tuned, specificando però che la generazione dei set di concetti non ha avuto un esito accettabile; sarebbe opportuno effettuare ulteriori studi e provare altre metodologie per ottenere set più coerenti e adatti al contesto.

Tutto il lavoro svolto può essere visualizzato ed eseguito dal seguente link: https://drive.google.com/file/d/1Xv8UuVDtrDl1G4XnvsjXxpabzfKe3QeC/, in formato IPython Notebook.

Conclusioni

Questa tesi ha presentato il processo di design, sviluppo e validazione di un applicativo mobile per fornire un supporto all'apprendimento a persone con difficoltà comunicative. L'app può essere utilizzata in autonomia per migliorare le proprie abilità, ma anche in compagnia, come strumento alternativo di dialogo. Inoltre, non risulta utile solo al diretto interessato, ma anche ai suoi tutor, che possono utilizzarla per rinforzare e semplificare l'apprendimento, assistendo alla lettura dei libri da loro forniti, insegnando come formulare correttamente le frasi dalla pagina *Leggiamo* e assegnando esercizi e giochi da svolgere in autonomia dalle pagine *Giochiamo* e *Diario*.

L'applicativo è stato realizzato con le più recenti tecnologie di sviluppo in ambito web e mobile, ed è attualmente online su un'infrastruttura che non necessita di alcun costo di mantenimento, un'importante criterio definito in fase di design. Durante lo sviluppo sono state proposte varie applicazioni dell'IA come supporto all'apprendimento per persone con difficoltà, sviluppando per ciascuna uno strumento in grado di integrarla e presentarla all'utente. Il risultato finale ha rispettato tutti i criteri definiti in fase di design e soddisfatto le aspettative degli esperti di dominio coinvolti.

Il core del sistema PictoAI può essere considerato un framework, con la possibilità di integrare facilmente nuove funzionalità o variazioni a quelle esistenti, in modo da migliorare ulteriormente le sue potenzialità educative e la sua capacità di adattarsi alle esigenze del singolo utente.

Tra gli sviluppi futuri dell'applicativo si può pensare a un'integrazione con LLM (Large Language Model), come ChatGPT, per dare più importanza alla componente Companion. Il Companion può diventare una sorta di "insegnante", in grado di supportare l'utente quando lavora in autonomia, reagendo coerentemente alle sue azioni, fornendo consigli quando richiesto e imparando i suoi punti di forza e lacune, dando anche la possibilità di generare i giochi della pagina *Giochiamo* di conseguenza. Un futuro rilascio dell'applicativo può portare alla creazione di una community numerosa, in grado di fornire un feedback sulle funzionalità offerte, ma soprattutto idee e consigli su possibili funzionalità da implementare. Una simile community gioverebbe enormemente a un applicativo Open come PictoAI: si potrebbe pensare all'introduzione di

un supporto a giochi creati da terzi per la pagina *Giochiamo*, dando l'accesso ad un catalogo di giochi creati dalla community in modo da permettere ad ogni utente di trovare qualcosa che stimoli il proprio interesse e che sia adatto alle proprie esigenze. Altrettanto interessante sarebbe un catalogo pubblico di libri generati dalla community, semplificandone il reperimento da parte di tutti gli utenti.

In conclusione, con un supporto costante alla piattaforma e l'implementazione di nuove ed efficaci funzionalità, PictoAI ha il potenziale per diventare uno strumento riconosciuto ed ampiamente utilizzato da tutta la comunità che si occupa dell'educazione di persone con difficoltà nella comunicazione.

Ringraziamenti

Alla Prof.ssa Carbonaro, al Prof. Moro e al Dott. Frisoni per avermi dato questa opportunità ed avermi assistito durante il suo sviluppo.

Alla mia famiglia, Aneta, Marcin e Natalia, per avermi permesso di arrivare dove sono adesso, supportandomi sempre al meglio delle proprie capacità.

Infine, un grazie anche a tutti i miei amici e cari, in particolar modo a Mattia, Gianni e Luca, per essermi sempre stati vicini, per il supporto dimostrato durante i momenti di difficoltà e per tutte le risate, che mi hanno permesso di finire con serenità questo lungo percorso.

Bibliografia

[1] Yasmin Elsahar, Sijung Hu, Kaddour Bouazza-Marouf, David Kerr, and Annysa Mansor. Augmentative and alternative communication (aac) advances: A review of configurations for individuals with a speech disability. *Sensors*, 19(8), 2019.

[2] American Speech-Language-Hearing Association (ASHA). Augmentative and alternative communication (aac).

[3] Gobierno de Aragón. Arasaac.

[4] Sophie Wodzak Duolingo. 3 ways duolingo improves education using ai.

[5] Jayr Alencar Pereira, David Macêdo, Cleber Zanchettin, Adriano Lorena Inácio de Oliveira, and Robson do Nascimento Fidalgo. Pictobert: Transformers for next pictogram prediction. *Expert Systems with Applications*, 202:117231, 2022.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] New Jersey 08544 USA Princeton University. Wordnet.

[8] Brian Macwhinney. The childes project: tools for analyzing talk. *Child Language Teaching and Therapy*, 8, 01 2000.

[9] MongoDB. Mern stack explained.

[10] IBM StrongLoop and other contributors. Express.

[11] Inc. Meta Platforms. React native "learn once, write anywhere.".

[12] AssistiveWare B.V. Proloquo2go.

[13] PRC-Saltillo. Touchchat.

[14] HelpICare by DIDACARE srl. The grid 3.

[15] PRC-Saltillo. Lamp words for life.

[16] OPENLAB Asti ETS. Simcaa.

[17] Dream Oriented Limited. Leeloo aac - assistive cards.

[18] CoughDrop. Coughdrop - every voice should be heard.

[19] Spoken Inc. Spoken – tap to talk aac.

[20] Kaitatzis Georgios. Leo - aac autism speech aide.

[21] Andrea Colzi. Aac talking tabs.

[22] Theo Browne. T3 stack.

[23] Microsoft. Typescript: Javascript with syntax for types.

[24] Alex Johansson. trpc - move fast and break nothing. end-to-end typesafe apis made easy.

[25] Tailwind Labs. Tailwind css - rapidly build modern websites without ever leaving your html.

[26] Inc Prisma Data. Prisma | next-generation orm for node.js typescript.

[27] Zoltan Kochan. Fast, disk space efficient package manager - pnpm.

[28] 650 Industries Inc. Expo.

[29] Zeit. Vercel: Develop. preview. ship. for the best frontend teams.

[30] Inc PlanetScale. Planetscale: The world's most advanced database platform.

[31] Clerk Inc. Clerk | authentication and user management.

[32] Hugging Face Inc. Hugging face – the ai community building the future.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[34] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning, 2020.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[36] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge, 2018.

[37] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online, June 2021. Association for Computational Linguistics.

[38] Mohamed Gad-Elrab, Daria Stepanova, Trung-Kien Tran, Heike Adel, and Gerhard Weikum. Excut: Explainable embedding-based clustering over knowledge graphs. 11 2020.

[39] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[40] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015.

[41] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.