# Accessible Board Games App - Developer Guide

## Spis treści

1. Cel aplikacji
2. Biblioteki wykorzystywane i planowane
3. Architektura i obecne moduły
4. Checklista modułów (stan, testy, dokumentacja)
5. Wytyczne budowania aplikacji dla osób niewidomych i niedowidzących
6. UI i flow aplikacji (do edycji w draw.io)
7. JSONy i dane o grach
8. Plan rozwoju modułów zarządzania grami
9. Peer-to-Peer i rozgrywka lokalna
10. Flow NFC

## 1. Cel aplikacji

**Dla kogo:**

* Osoby **niewidome**
* Osoby **niedowidzące**
* Opiekunowie / nauczyciele pracujący z osobami niewidomymi

**Obsługiwane tryby (planowane):**

* Tryb **czarno-biały** (duży kontrast, brak animacji)
* Tryb **wysokiego kontrastu** (zgodny z WCAG)
* Tryb **głosowy + gesty** (dla niewidomych)
* Tryb **współgracza** (peer-to-peer dla opiekuna / innej osoby widzącej)

## 2. Biblioteki wykorzystywane i planowane

### Aktualne:

* **Kivy** – budowa UI, obsługa ekranów
* **Pyjnius** – dostęp do funkcji Androida (TTS, NFC)
* **Python TextToSpeech / Android TTS** – odczyt komunikatów
* **json** – przechowywanie stanu gier i szablonów

### Planowane:

* **Kivy Gesture Behavior** – obsługa gestów
* **KivyMD** – dla łatwiejszej implementacji trybu kontrastowego
* **Bluetooth / socket / zeroconf** – peer-to-peer
* **OCR (np. pytesseract)** – do automatycznego rozpoznawania kart (opcjonalnie)

### Systemy docelowe:

* **Android** (priorytet)
* **iOS** (docelowo) – wymaga przemyślenia:
  + Zastąpienie Pyjnius (działa tylko na Androidzie)
  + Integracja z **VoiceOver** i systemowym TTS

**Wyzwania:**

* Różnica obsługi NFC, gestów i TTS pomiędzy Androidem i iOS
* Potrzeba stworzenia abstrakcyjnej warstwy dostępności dla wielu systemów

## 3. Architektura aplikacji i obecne moduły

| Moduł | Lokalizacja | Opis |
| --- | --- | --- |
| main.py | korzeń projektu | Inicjalizacja, ScreenManager, StartupSettings |
| startup\_settings\_applier | startup/ | Wczytuje config i ustawia TTS, język, głośność |
| logic/commands | logic/commands/ | Akcje np. ustawienie głosu, zmiana szybkości TTS |
| logic/queries | logic/queries/ | Pobieranie danych z config, gier itp. |
| navigation | navigation/ | Obsługa przełączania ekranów (ScreenManager) |
| screens/settings/ | UI ustawienia aplikacji |  |
| services/tts | services/tts/ | Warstwa do komunikacji z systemowym TTS |

Planowane:

* nfc\_mapper/ – przypisywanie tagów do kart
* game\_assistants/ – wsparcie rozgrywki głosem
* peer\_connection/ – tryb współgracza, synchronizacja stanu
* gesture\_handler/ – interakcje gestami
* ocr\_reader/ – moduł OCR do rozpoznawania tekstu ze zdjęć kart

Struktura katalogów gier:

data/  
└── tajniacy/  
 ├── template.json # szablon kart do mapowania  
 ├── info.json # dane o grze  
 ├── instructions.json # instrukcja (skrócona)  
 └── user\_game\_data.json # zmapowane karty (użytkownik)  
  
game\_assistants/  
├── tajniacy.py  
├── smoki.py  
├── sojusz\_na\_juz.py  
└── wspolne.py

### Komunikacja game\_assistants/\*.py z JSON:

* template.json: zawiera schemat pól, jakie użytkownik powinien uzupełnić podczas mapowania (np. A - tytuł, B - opis, C - działanie). tajniacy.py używa go, aby wiedzieć, jakie dane może odczytać z karty i jakie akcje są dostępne.
* instructions.json: instrukcja do gry w formacie JSON podzielonym na sekcje tematyczne. Moduł asystujący może odczytać głosowo fragmenty instrukcji na życzenie użytkownika.
* user\_game\_data.json: przechowuje dane wprowadzone przez użytkownika podczas mapowania kart – unikalne ID tagu NFC oraz dane z OCR
* Moduł tajniacy.py interpretuje zawartość odczytaną z tagu NFC (znajdując ją w user\_game\_data.json) i na tej podstawie generuje komunikaty, dostępne akcje i decyzje głosowe.

## 4. Checklista modułów

W oddzielnym pliczku: docs/module\_checklist.xlsx)

## 5. Wytyczne budowania aplikacji dla osób niewidomych/niedowidzących

* **UI liniowe**: wszystkie ekrany są jednoetapowe, posiadają wyraźne komunikaty i przyciski “kontynuuj / wróć / wyjście”
* **Potwierdzenie głosowe**: każda interakcja wywołuje natychmiastowy komunikat TTS
* **Gesty uproszczone**:
  + 1x klik = odczytaj
  + 2x klik = zatwierdź / przejdź dalej
  + Long press = powtórz
* **Zdarzenia systemowe**: np. przy połączeniu telefonicznym
  + pauza w grze (freeze)
  + opcja odebrania / odrzucenia połączenia za pomocą przycisków
* **Brak wymogu logowania**: dla wygody i niezależności użytkownika
* W przypadku sesji lokalnych pojawia sie ekran do wpisania/ podania nazwy gracza
* Logowanie dla nauczycieli/instytycji schowane w ustawieniach - dostęp do vip opcji, dla subskrybentów instytucjonalnych
* **Dźwięki i wibracje**: opcjonalnie jako wzmocnienie informacji
* **Pamięć sesji**: aplikacja zapamiętuje kontekst (np. mapowanie kart), aby można było kontynuować

## 6. UI i flow aplikacji (do edycji w draw.io)

Przykładowy flow:

Main Menu  
 |  
 |-- [1] Katalog gier  
 | |-- Gra 1  
 | | |-- Informacje o grze  
 | | |-- Lokalne sesje rozgrywki  
 | | |-- Wprowadzanie gry (4 etapy)  
 | | | |-- Czy masz fizyczną wersję? tak/ nie  
 | | | |-- Czy masz {X} znaczników NFC? tak/nie  
 | | | |-- Naklej znacznik i przyłóż kartę   
 | | | |-- Zrób zdjęcie

| |-- Gra 2  
 | | |-- Informacje o grze  
 | | |-- Lokalne sesje rozgrywki  
 | | |-- Wprowadzanie gry (4 etapy)  
 | | | |-- Czy masz fizyczną wersję? tak/ nie  
 | | | |-- Czy masz {X} znaczników NFC? tak/nie  
 | | | |-- Naklej znacznik i przyłóż kartę   
 | | | |-- Zrób zdjęcie

| |-- Gra 3  
 | | |-- Informacje o grze  
 | | |-- Lokalne sesje rozgrywki  
 | | |-- Wprowadzanie gry (4 etapy)  
 | | | |-- Czy masz fizyczną wersję? tak/ nie  
 | | | |-- Czy masz {X} znaczników NFC? tak/nie  
 | | | |-- Naklej znacznik i przyłóż kartę   
 | | | |-- Zrób zdjęcie   
 |  
 |-- [2] Moje gry  
 | |-- Gra 1  
 | | |-- Informacje o grze  
 | | |-- Instrukcja gry (skrócona)  
 | | |-- Rozpocznij grę (moduły TTS, gesty)  
 | | |-- Ustawienia gry (co czytać po skanowaniu)

|-- [3] Ustawienia aplikacji  
 | |-- Głos / język / Tryb kontrastowy / gesty / TTS  
 | |-- Logowanie  
 | |-- Exit

## 7. JSONy i dane o grach

* template.json: szablon na podstawie instrukcji wydawcy
* user\_game\_data.json: uzupełniony przez użytkownika plik z kartami
* info.json: dane informacyjne o grze
* instructions.json: skrócona instrukcja krok po kroku

**Opis pól:**

### template.json

{  
 "game\_title": "Tajniacy",  
 "publisher": "Rebel",  
 "total\_cards": 60,  
 "fields": {  
 "A": "title",  
 "B": "description",  
 "C": "type"  
 },  
 "card\_distribution": {  
 "agent": 20,  
 "bystander": 30,  
 "assassin": 10  
 },  
 "required\_modules": ["card\_reader", "gesture\_handler", "tts\_feedback"]  
}

### user\_game\_data.json

{  
 "cards": [  
 {  
 "nfc\_uid": "ABCD1234",  
 "title": "Słoń",  
 "description": "Zwierzę duże",  
 "type": "agent"  
 }  
 ],  
 "mapping\_progress": {  
 "total": 60,  
 "mapped": 1  
 }  
}

### info.json

{  
 "game\_title": "Tajniacy",  
 "publisher": "Rebel",  
 "description": "Gra słowna polegająca na zgadywaniu agentów...",  
 "tags": ["dedukcja", "słowna", "kooperacyjna"],  
 "instructions\_path": "./instructions.json",  
 "accessibility\_modes\_supported": ["voice", "gesture", "contrast"],  
 "players": "2-4",  
 "estimated\_play\_time": "30 min",  
 "difficulty": "średnia"  
}

## 8. Plan rozwoju modułów zarządzania grami

### Moduły:

* game\_assistants/<gra>.py: logika głosowa dla danej gry
* instruction\_explainer/: odczytywanie instrukcji krok po kroku
* nfc\_mapper/mapper.py: przypisywanie UID znaczników do kart z template

### Powiązania:

* mapper → korzysta z template.json → tworzy user\_game\_data.json
* game\_assistants → korzysta z danych o kartach → reaguje na UID i odczytuje zawartość
* peer\_connection → współdzieli stan gry (opcjonalnie synchronizacja kart / efektów)

### Kolejność prac:

1. Stworzenie template JSON (instrukcja → pola do uzupełnienia)
2. Moduł nfc\_mapper – proces parowania UID ↔ dane z template
3. Interfejs do kontynuacji mapowania (jeśli przerwane)
4. Dodanie instruction\_explainer – sekwencyjne instrukcje
5. Implementacja game\_assistants (dla Tajniacy, Smoki, Sojusz na Już)

## 9. Peer-to-Peer i rozgrywka lokalna

### Scenariusz:

* Niektóre gry wymagają trybu lokalnego – po kliknięciu “Rozpocznij grę” pojawia się komunikat:

“Celem zagrania w ten tytuł musisz utworzyć rozgrywkę lokalną. Nadaj teraz nazwę dla tej sesji – inni użytkownicy będą mogli w ten sposób znaleźć tę rozgrywkę.”

* Następnie ekran: “Podaj nazwę gracza”
* Gracze przechowują dane sesji lokalnie (np. UID gracza, przypisane karty itp.)
* Pozostali gracze widzą dostępne rozgrywki po wejściu w katalog gier → wybranie gry → lista lokalnych sesji

### Moduły:

* peer\_connection/session.py – hostowanie gry i łączenie się przez Bluetooth / Wi-Fi
* peer\_connection/sync.py – synchronizacja stanu (np. odkryte karty)
* peer\_connection/discovery.py – wykrywanie sesji lokalnych w pobliżu

### Ekrany:

* Stwórz rozgrywkę: nazwa sesji
* Podaj nazwę gracza
* Lobby: oczekiwanie na graczy
* Rozgrywki lokalne: lista sesji dostępnych w danym tytule

### Tryb nauczyciela / instytucji:

* Ekran logowania dostępny w ustawieniach
* Po zalogowaniu: pojawia się nowa sekcja “Tryb nauczyciela”
* Możliwość zarządzania listą uczniów (lokalnie?) – do sprawdzenia pod kątem legalności
* Dodatkowe funkcje: statystyki rozgrywek, eksport wyników

10. Flow NFC

Ekran 1: Czy masz fizyczną wersję gry?

Ekran 2: Czy masz X znaczników? (z template.json)

Ekran 3: Przyłóż kartę i przyklej znacznik NFC

Ekran 4: Zrób zdjęcie – OCR → przypisz dane do pól A/B/C (obowiązkowe)

Ekran 5: Powtarzaj proces do zmapowania wszystkich kart

Ekran 6: Jeśli znacznik jest już zmapowany → komunikat + przejście dalej

Ekran 7: Jeśli przerwano proces, możliwość kontynuacji (z mapping\_state\_tracker)