Introducere

Ce realizează proiectul?

În urma experienței personale în domeniul cititului online, am ales să dezvolt o aplicație care facilitează cititul și oferă o experiență completă utilizatorului. În acest sens, nume-app oferă posibilitatea de a descoperi noi cărți în funcție de propriile interese, de a ține evidența atât a cărților citite curent, cât și celor finalizate, dar și o planificare în viitor a acestora. Mai mult decât atât, fontul și fundalul cărților poate fi complet customizabil, iar în timpul cititului, utilizatorul are posibilitatea să activeze opțiunea de scroll automat în funcție de gesturi, dar și să asculte melodiile reprezentative pentru capitol, propuse de autorul cărții. În nume-app, oricine se simte inspirat poate deveni autor al unei cărți, și poate urmări reacțiile cititorilor săi prin secțiunea de comentarii aferentă fiecărui paragraf.

De ce (contextul problemei, impact, motivație)?

Motivația alegerii temei a venit după utilizarea unei aplicații deja existente pe piață. Odată cu apariția stand-urilor pentru telefoane și tablete, am considerat că mișcarea de a da pagina poate fi ușor automatizată de către camera telefonului, care acum poate sta stabil, prin urmărirea poziției retinei utilizatorului, care poate lua o poziție comodă mai departe de ecranul telefonului. Această nouă îmbunătățire face ca cititul unei cărți să se asemene mai mult cu vizionarea unui film, în care, de asemenea, cadrele se desfășoară automat. În plus, am considerat necesar ca toate featurile aplicației să se afle la un click distanță, așa că nume-app oferă posibilitatea de a încărca si asculta melodii în timpul cititului, fără a le mai deschide dintr-o aplicație auxiliară de streaming, cum se întâmplă în prezent pe aplicațiile de pe piață.

State of the art

Aplicația care m-a inspirat să implementez acest proiect se numește Wattpad, iar conceptul de bază al nume-app provine de aici. Ca și aplicații similare deja prezente pe piață pot enumera bine cunoscutul Kindle Amazon, care dispune și de aferentul ebook reader, dar și Bookly.

*Tab. 1 – Analiza comparativă a aplicațiilor similare de pe piață*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caracteristici** | **Bookly** | **Kindle amazon** | **Wattpad** | **MyApp** |
| Link store | [GooglePlay](https://play.google.com/store/apps/details?id=com.twodoor.bookly&hl=ro&gl=US&pli=1) | [GooglePlay](https://play.google.com/store/apps/details?id=com.amazon.kindle&hl=en_US) | [GooglePlay](https://play.google.com/store/apps/details?id=wp.wattpad&hl=en_US) | - |
| Notă store | 4.3 / 5 | 4.8 / 5 | 4.4 / 5 | - |
| Nr. instalări | 100K+ | 100M+ | 100M+ | - |
| Nr. ratings | 4.79K | 3.46M | 4.79M | - |
| Adds / in-app purchases | - | x | x | - |
| Creare cont | - | x | x | x |
| Targete de citit zilnice/lunare/anuale | x | x | - | - |
| Vizualizarea de statistici pentru timpul petrecut citind | x | x | - | - |
| Discută cu AI-ul aplicației despre cărți | x | - | - | - |
| **Caracteristici** | **Bookly** | **Kindle amazon** | **Wattpad** | **MyApp** |
| Îmbunătățirea vocabularului prin „cuvântul zilei” | x | - | - | - |
| Importă o carte deja existentă online | x | x | - | - |
| Importă un PDF personal pe care să îl citești | x | - | - | - |
| Scanează un ISBN cod de pe o carte fizică și importă conținutul în aplicație | x | - | - | - |
| Descoperă cărți în funcție de gen | - | x | x | x |
| Recomandări de cărți | - | x | x | x |
| Permite acordarea de recenzii unei cărți | - | x | x | - |
| Permite conversațiile cu alți utilizatori | - | x | x | x |
| Permitea evidențierea unor pasaje din carte | - | x | - | - |
| Arată locația curentă în carte | - | x | x | - |
| Învață viteza de citire a utilizatorului | - | x | - | - |
| Poate fi folosit pe Kindle | - | x | - | - |
| Permite citirea offline a cărților | - | - | x | - |
| Oferă notificări cu privire la actualizarea capitolelor | - | x | x | x |
| Permite ascultarea muzicii în aplicație | - | - | - | x |
| Permite personalizarea modului de citire | x | x | x | x |
| Permite planificarea în viitor a cărților | - | x | - | x |
| Scroll automat în funcție de privire | - | - | - | x |

Design și implementare

La baza aplicației stau cele 2 directoare aferente front-endului și back-endului. La rândul lui, frontend-ul se împarte în subdirectoarele „ios” și „android”. A fost nevoie de această împărțire datorită unor aspecte care funcționau doar pe sistemul de operare Android. În ambele cazuri, interfața cu utilizatorul a fost programată în limbajul Typescript, cu ajutorul framework-urilor React Native și Expo. În ceea ce privește backend-ul, acesta a fost programat în Java, mai exact Spring Boot, și urmărește arhitectura bazată pe nivele: controllere, servicii și modele bazei de date. Baza de date folosită este Firestore Database pentru salvarea datelor despre utilizatori și cărți, și Firestore Storage pentru salvarea imaginilor reprezentând coperțile cărților, cât și ale altor funcționalități ce urmează a fi prezentate.

O prezentare sumară a decursului datelor în aplicație ar putea fi descrisă astfel: în primă instanță sunt afișate componentele din interfața grafică, care au afiliate funcții care se declanșează odată cu anumite acțiuni (callbacks). Aceste funcții sunt deseori asincrone, și apelează serviciile care, la rândul lor adresează diferite cereri server-ului de backend, la o anumita adresă. Aici vor fi chemate implicit funcțiile „controller”, care au rolul de a mapa o procedură mai specifică la adresa care a fost invocată. Această procedură specifică este serviciul din backend, unde se tratează în principal acumularea și prelucrarea datelor provenite din baza de date. Și nu în ultimul rând, modelele reprezintă clase Java care modelează o anumită entitate necesară în logica programului (ca de exemplu utilizatorul, sau cartea).

În cele ce urmează voi prezenta particularități ale ecranelor implementate în aplicație.

1. Ecranul „Login și Register”

Prima întâlnire pe care utilizatorul o are cu aplicația sunt ecranele de Login și Register. Acestea au fost combinate pentru a apărea în același ecran, folosind conceptul de „conditional rendering” prezent în React Native. Pentru comunicarea cu backend-ul s-au folosit thread-uri asincrone, implementate cu ajutorul async/await. În ceea ce privește codificarea parolei, s-a folosit librăria BCrypt prezentă în Spring Boot, mai exact metodele de generare salt (un șir de caractere unic pentru fiecare utilizator, folosit în encodarea parolei) și hash-password.

Imediat după ecranul „Register” urmează un ecran în care utilizatorul este rugat să selecteze genurile de cărți favorite. Acesta a fost implementat ca și un Grid Layout, folosind componenta de Checkbox din Expo, urmând ca, în baza de date să fie salvată o listă cu genurile selectate.

1. Ecranul „Home”

Ecranul „Home” este punctul de start al aplicației. Aici sunt afișate sugestii de cărți bazate pe interesele utilizatorului, mai exact cărți al căror gen se regăsește în lista de preferințe a utilizatorului. De asemenea, sunt afișate cărțile cu cel mai mare număr de cititori, cât și cărțile recent citite. O primă operațiune care se face imediat după lansarea ecranului este aducerea datelor necesare din backend pentru a preveni întârzieri la lansarea altor ecrane, care, la rândul lor, se folosesc de aceste date. Ca de exemplu, cărțile din rândul cărților citite curent vor fi afișate atât aici, cât și în secțiunea dedicată din ecranul „Bibliotecă”.

1. Ecranul „Bibliotecă”

Ecranul „Bibliotecă” conține 3 secțiuni: cărți citite curent, cărți finalizate, și un planificator pe luni a cărților. Primele 2 secțiuni se bazează pe afișarea într-un Grid a unor cărți deja aduse din baza de date și salvate local. Cea de a treia secțiune are la baza implementării conceptul de Drag and Drop, implementat cu ajutorul librăriei „react-native-gesture-handler”. Sunt afișate într-o listă cadrane specifice fiecărei luni din an, împreună cu coperțile cărților planificate în luna respectivă. Apăsarea pe butonul „Edit” conduce la o altă pagină, împărțită în două secțiuni: o listă cu derulare orizontală care conține cărți din rândul celor citite curent, dar care nu au fost încă planificate, și un cadran specific lunii din an care este editată. Prima secțiune conține o entitate specială de carte, și anume DraggableBook, care are la bază un cadran special, și anume PanGestureHandler. PanGestureHandler, după cum îi spune și numele, este o vedere specială care permite un control mai detaliat al gesturilor, și anume informații despre începutul gestului, când gestul este activ, și când s-a terminat. Gestul devine activ doar după apăsarea îndelungată pe pictograma unei cărți, iar apoi, urmărește traseul degetului (coordonatele x și y ale cărții se translatează liniar). Când degetul este ridicat, se verifică poziția cărții și se compară coordonatele cu cele ale cadranului cu rol de „Drop”. În cazul în care pictograma a fost plasată în afara cadranului, aceasta se întoarce la coordonatele inițiale. Pentru aceasta, s-a folosit biblioteca 'react-native-reanimated', ca tranziția imaginii din punctul în care a fost lăsată și până la coordonatele inițiale să decurgă cursiv. În cazul în care pictograma a fost plasată în cadranul lunii curente, urmează a fi apelate o serie de funcții. De menționat este faptul că gesturile descrise anterior rulează explicit în thread-ul interfeței grafice, iar funcțiile apelate care gestionează adăugarea/ștergerea unei cărți rulează în thread-ul principal de execuție Javascript. Aceste funcții fac apel la serviciile backend care actualizează datele existente, și adaugă sau șterge local o entitate din lista cărților planificate. Această listă este declarată cu ajutorul unei variabile de stare (hook-ul useState prezent în React Native), și asta face ca schimbarea ei să declanșeze o re-afișare cu noile valori. O nouă entitate a cărții este folosită în ecranul de Edit, și anume EditableBook, care are particularitatea că semnul „-”, aferent ștergerii, este prezent deasupra pictogramei cărții. EditableBook și DraggableBook sunt componente separate, care comunică cu ecranul Edit prin React Native Props.

1. Ecranul de citit

Acest ecran se lansează în momentul în care utilizatorul apasă pe pictograma unei cărți. Începând cu funcționalitatea de bază, ecranul afișează conținutul unui capitol dintr-o carte, încărcat printr-un serviciu extern. Conținutul capitolului este format prin înlănțuirea paragrafelor sale, astfel: din baza de date se extrage un vector de dicționare în care sunt prezente id-urile și conținutul paragrafelor. Conținutul unui paragraf, împreună cu un buton cu iconița unui comentariu sunt adăugate împreună într-un cadru. Apoi, se calculează o distribuire a textului. Navigarea între pagini este orizontală, deci în fiecare pagină (cadru) trebuie distribuite atâtea paragrafe câte încap în întregime în funcție de mărimea fontului. După distribuirea textului, se poate face afișarea componentelor. Paginile sunt incluse într-o structură de date specifică React Native, și anume FlatList. Această abordare asigură o navigare lină în cadrul capitolului și o încărcare eficientă a textului pe măsură ce utilizatorul avansează prin lectură. De asemenea, aceasta permite declanșarea de evenimente pentru scroll, cât și un scroll programabil, necesare pentru funcționalitatea de scroll cu ajutorul gesturilor.

Pentru a îmbunătăți experiența vizuală, utilizatorul are posibilitatea de a personaliza aspectul textului prin intermediul unei componente de tip „BottomSheet”. Aceasta include opțiuni pentru schimbarea culorii de fundal, tipului de font și a dimensiunii acestuia. Schimbarea dimensiunii fontului declanșează o recalculare a modului în care textul este aranjat pe ecran, asigurându-se că se potrivește corect în spațiul disponibil, indiferent de mărimea selectată. BottomSheet se poate declanșa prin tragerea în sus a bării din partea de jos a ecranului. Această bibliotecă oferă o modalitate de a adăuga straturi suplimentare de personalizare, fără a încărca interfața utilizatorului.

Navigarea în text este facilitată prin gesturi, cu ajutorul bibliotecii „Face Detector”, prezentă în framwork-ul Expo. Activând această funcție din BottomSheet, utilizatorul poate trece la pagina următoare sau precedentă prin simpla înclinare a capului. Detectarea mișcării capului se face prin monitorizarea unghiurilor Yaw și Roll. „Yaw angle” se referă la rotația feței pe axa orizontală, în timp ce „Roll angle” indică înclinarea capului pe axa orizontală. Aceste două unghiuri sunt folosite pentru a determina direcția în care utilizatorul își înclină capul. De exemplu, o înclinare la dreapta sau la stânga va fi detectată ca o schimbare în sensul pozitiv sau negativ al „Roll angle”, în timp ce, ca metodă de filtrare, „Yaw Angle” se menține aproximativ constant, indicând o poziție dreaptă a capului (ne-rotită). Aplicația interpretează aceste schimbări de unghi ca semnale pentru navigarea în text. De exemplu, o înclinare la dreapta semnifică trecerea la pagina următoare, în timp ce o înclinare la stânga înseamnă revenirea la pagina anterioară. Detectarea se face în mod regulat, la un interval de 500 milisecunde. Acest lucru asigură că aplicația răspunde în timp real la mișcările utilizatorului, fără a consuma resurse excesive.

Pe lângă acestea, ecranul oferă acces la un meniu de cuprins, unde capitolele cărții sunt enumerate și accesibile printr-un singur click. Acest lucru facilitează navigarea rapidă între diferite secțiuni ale cărții.

Utilizare

La deschiderea aplicației, utilizatorul este întâmpinat de paginile de autentificare, cu opțiunile de a se loga sau de a se înregistra. Fiecare buton de pe ecranele de "Login" și "Register" este creat cu TouchableOpacity sau TouchableHighlight, care reacționează vizual la interacțiunea utilizatorului. Atunci când butonul este atins, opacitatea sau culoarea de fundal se schimbă pentru a indica faptul că a fost apăsat, iar un callback onPress este declanșat pentru a procesa acțiunea cerută de utilizator, fie aceea de a se autentifica sau de a crea un cont nou.

După înregistrare, utilizatorul este îndrumat să selecteze interesele sale, ajutând aplicația să personalizeze conținutul recomandat. Această pagină folosește checkbox-uri care permit utilizatorului să selecteze sau să deselecteze opțiunile cu un simplu click. Fiecare casetă de selectare este de asemenea legată de un callback onPress, care schimbă starea elementului din ne-selectat în selectat și invers, actualizând starea internă a aplicației pentru a reflecta preferințele utilizatorului.

După finalizarea selecției de interese, utilizatorul este redirectat către pagina principală „Home”. Utilizatorul poate naviga prin secțiunile de cărți, care sunt prezentate sub forma unui carusel de cărți. Acesta e implementat cu ajutorul unui ScrollView, care se bazează pe un gest de glisare. Fiecare carte poate fi selectată prin atingere, ducând la o pagină de prezentare a cărții respective, dacă aceasta nu se află deja în bibliotecă. Aici, utilizatorul regăsește coperta, descrierea cărții prin care poate naviga tot cu scroll, cât și un buton de „Adaugă în bibliotecă”, implementat cu ajutorul TouchableOpacity, dar și un buton de navigare către conținutul cărții.

Navigând prin meniul aplicației, utilizatorul găsește secțiunea "Library", care este organizată în subsecțiuni: "Current Readings", "Finalized Readings" și "Reading Planner". În "Reading Planner", utilizatorul vede afișată o listă cu lunile anului curent, cu câte o imagine de fundal reprezentativă, care conține pictograme ale cărților deja planificate. Fiecare cadru al unei luni dispune de un buton de edit, care, prin apăsare, duce la pagina de Edit Reading Planner. Aici utilizator va folosi funcționalitatea de drag and drop, care rulează astfel: se apasă lung pe o carte pentru a o selecta, după care se poate trage în cadrul care indică luna. Imaginea cărții revine la poziția originală dacă s-a tras oriunde în afara cadrului care indică luna. Acest proces de drag and drop este gestionat printr-un sistem de coordonate care actualizează poziția cărții pe ecran în timp real, urmărind fidel gesturile utilizatorului.

Când o carte este selectată din oricare dintre aceste secțiuni, utilizatorul este redirecționat către ecranul de citit, unde poate folosi gestul de swipe pentru a naviga printr-un scroll orizontal care derulează paginile cărții. Aici, gesturile tactile pot fi înlocuite de gesturi ale capului, dacă funcționalitatea de „Gesture Scrolling” a fost activată. Aceasta se activează prin acționarea unui buton de tipul „switch”. În partea de sus a ecranului, se poate accesa meniul de cuprins prin apăsarea iconiței sub formă de triunghi. Se va deschide un ecran nou, în care utilizatorul poate naviga prin lista de capitole ale cărții. Fiecare nume de capitol este o componentă Touchable, și, astfel conduce la ecranul care prezintă conținutul capitolului.

În partea de jos a ecranului de citit, utilizatorul poate să tragă un panou din partea de jos a ecranului spre partea de sus pentru a dezvălui mai multe opțiuni sau informații. Aici sunt prezente următoarele: un switch pentru activarea „Gesture Scrolling”, vederi care au diferite culori de fundal din care utilizatorul poate alege apăsându-le. Odată apăsată o vedere, aceasta rămâne selectată până când o alta este apăsată. Mai sunt disponibile butoane de tipul Touchable Opacity, care, prin apăsare repetată, cresc sau micșorează dimensiunea fontului care este afișat pe ecran. De asemenea, mai este prezentă o casetă în care e afișat numele fontului scrisului afișat. Un click pe această casetă declanșează apariția unei liste are conține nume de fonturi disponibile. În mod similar, conținutul capitolului va fi afișat cu fontul selectat.

Concluzii

În procesul de dezvoltare și testare a aplicației, am întâmpinat diverse provocări și rezultate neașteptate care au redirecționat cursul proiectului și au deschis noi posibilități.

Inițial, aplicația a fost concepută pentru utilizarea pe iOS. Cu toate acestea, s-a descoperit că biblioteca „Face Detector”, esențială pentru funcționalitatea de navigare prin gesturi, nu era disponibilă pe iOS în versiunile de dezvoltare (development builds) ale Expo. O dificultate întâmpinată în dezvoltarea aplicației a fost că Expo, framework-ul utilizat, vine cu un pachet predefinit de API-uri și funcționalități. Pentru a include biblioteci native suplimentare, precum cea necesară pentru detectarea feței, ar fi necesar să se renunțe la mediul gestionat de Expo. Ca soluție temporară, am achiziționat un dispozitiv Android pentru a continua dezvoltarea, deoarece Android permite o mai mare flexibilitate în utilizarea bibliotecilor native.

Apoi, au fost testate mai multe tehnologii pentru a implementa navigarea prin gesturi:

MediaPipe: O bibliotecă puternică pentru procesarea stream-urilor multimedia, care include soluții de detectare a feței. Din păcate, aceasta nu mai poate fi instalată ca librărie gata de folosire, ci e disponibilă sub forma unui repository, care trebuie adaptat la cerințele personale ale aplicației. Astefel, necesită cunoștințe aprofundate pentru a fi integrată eficient într-un proiect Expo.

OpenCV: Este un standard în procesarea de imagini și oferă capabilități avansate, însă integrarea cu Expo este complexă și documentația specifică este limitată.

Google ML Kit: Oferă o serie de funcționalități legate de învățarea automată, inclusiv detectarea feței, dar nu există o integrare directă și simplă cu Expo.

TensorFlow Lite: Este o bibliotecă puternică pentru modele de învățare automată pe dispozitive mobile, însă documentația pentru integrarea într-un mediu Expo este limitată. Documentația disponibilă face referire la integrarea în aplicații Web React, iar transpunerea ei într-un proiect Expo React Native a dus la stoparea repetată a aplicației, și blocarea ei, întrucât componenta expo-camera rula separat, cu sarcina de a face câte o poza la un interval de 500 milisecunde, pentru a trimite poza ca parametru către modelul TensorFlow. Intervalul de 500 milisecunde este prea scurt pentru ca expo-camera să ofere input-ul așteptat, iar asta conduce la promisiuni începute, dar nerezolvate.

Rezultatele de la testare au evidențiat o diferență semnificativă între sistemele de operare: fonturile care funcționau pe iOS nu erau disponibile pe Android. Această descoperire va necesita alegerea unor fonturi alternative sau implementarea unui sistem de fonturi care să se adapteze în funcție de platforma utilizată.

Un aspect interesant și captivant a fost descoperirea unghiurilor măsurate pentru detectarea poziției feței. Explorarea unghiurilor „Yaw” și „Roll” mi-a permis o înțelegere mai bună a modului în care fața și trăsăturile umane sunt detectate.

În concluzie, procesul de dezvoltare a acestei aplicații mi-a oferit informații valoroase despre limitările și posibilitățile platformelor telefoanelor mobile, despre cum să navighez în jurul constrângerilor tehnice și să găsesc soluții pentru lipsurile descoperite, să învăț să gestionez lucrul cu diferite biblioteci, și mai ales, mi-a oferit o bucurie consistentă când ideea de la care s-a născut acest proiect s-a implementat cu succes.