Aplikacje mobilne

Zadanie do wykładu 2

Kino Pod Gwiazdami – ekrany z podstawowych komponentów

View, Text, Image, Pressable, ScrollView, FlatList, SectionList, Modal, Flexbox

Mateusz Pawełkiewicz

Wersja: październik 2026

Expo SDK 58 · React Native 0.88 · React 19.3 · TypeScript

Kod sprawdzony: tsc, ESLint, Jest; aplikacja uruchomiona w przeglądarce (react-native-web)

Cel zadania

Na kolejnych wykładach budujemy jedną, coraz bogatszą aplikację **„Kino Pod Gwiazdami”** – repertuar kina, seanse, wybór miejsc, bilety. Każde zadanie dodaje do niej to, o czym był wykład. W tym etapie tworzymy projekt i trzy widoki zbudowane wyłącznie z podstawowych komponentów React Native: **repertuar** (karuzela premier + filtrowana siatka filmów), **seanse** pogrupowane według dni oraz **plan sali** z wyborem miejsc.

Nawigacji jeszcze nie ma (to temat wykładu 3) – widoki przełączamy zwykłym stanem `useState` i własnym komponentem `SegmentedControl`. Dzięki temu cała uwaga skupia się na komponentach, stylach i listach.

> **Informacja**
> Gotowy kod tego etapu: katalog „kod/w02” obok instrukcji.

Co zbudujemy

- projekt Expo SDK 58 z TypeScriptem i aliasem importów `@/…`,
- własne fonty (Inter) ładowane przez `useFonts` i ekran powitalny (splash) ukrywany po załadowaniu fontów,
- jasny i ciemny motyw zależny od ustawień telefonu (`useColorScheme`) – kolory, odstępy i typografia w jednym pliku,
- karuzela premier: poziomy `ScrollView` z „przyklejaniem” kart (`snapToInterval`) i gradientem na zdjęciu,
- siatka filmów `FlatList` z liczbą kolumn zależną od szerokości ekranu (`useWindowDimensions`), wyszukiwarka `TextInput`, filtr `Switch` i „chipsy” gatunków,
- seanse w `SectionList` z przyklejanymi nagłówkami dni,
- plan sali zbudowany wyłącznie z Flexboxa (rzędy, przejścia, miejsca VIP) z wyborem miejsc i paskiem podsumowania nad dolną krawędzią ekranu (`useSafeAreaInsets`),
- szczegóły filmu w natywnym oknie `Modal` (arkusz na iOS),
- obrazy przez `expo-image` z rozmytym podglądem (blurhash) i płynnym pojawieniem się.

Co z dokumentacji pokazujemy w praktyce

Temat z wykładu / dokumentacji	Gdzie w kodzie
View, Text, StyleSheet, Flexbox (flexDirection, gap, flexWrap, aspectRatio)	`SeatMap.tsx`, `MovieCard.tsx`, `theme.ts`
Pressable: stan `pressed`, `android_ripple`, `hitSlop`	`MovieCard.tsx`, `SeatMap.tsx`, `GenreChips.tsx`
ScrollView poziomy: `snapToInterval`, `decelerationRate`	`PremiereCarousel.tsx`
FlatList: `numColumns`, `keyExtractor`, `ListHeaderComponent`, `initialNumToRender`	`RepertoireView.tsx`
SectionList: sekcje, `stickySectionHeadersEnabled`, `ItemSeparatorComponent`	`ScreeningsView.tsx`
Modal: `animationType`, `presentationStyle`, `onRequestClose`	`MovieDetailsModal.tsx`
TextInput, Switch (kontrolowane przez stan)	`RepertoireView.tsx`

Temat z wykładu / dokumentacji	Gdzie w kodzie
useWindowDimensions – responsywny układ	`RepertoireView.tsx`, `PremiereCarousel.tsx`
Platform.OS – różnice iOS/Android	`MovieCard.tsx`
SafeAreaProvider / SafeAreaView / useSafeAreaInsets	`App.tsx`, `HallView.tsx`
expo-image (placeholder, contentFit, transition), expo-linear-gradient	`MovieCard.tsx`, `PremiereCarousel.tsx`
expo-font + @expo-google-fonts, expo-splash-screen	`App.tsx`
useColorScheme – tryb ciemny	`theme.ts`
memo / useMemo / useCallback – mniej zbędnych renderów	`SeatMap.tsx`, `RepertoireView.tsx`
Dostępność: `accessibilityRole`, `accessibilityLabel`, `accessibilityState`	`SeatMap.tsx`, `SegmentedControl.tsx`

Przed rozpoczęciem

- Node.js 22 LTS (lub nowszy) i edytor (VS Code),
- telefon z aplikacją **Expo Go** (w tej samej sieci Wi-Fi co komputer) albo emulator Androida / symulator iOS,
- konto Expo nie jest potrzebne na tym etapie.

Instalacja pakietów

Tworzymy projekt z szablonu TypeScript i doinstalowujemy biblioteki. **Zawsze używaj `npx expo install`** zamiast samego `npm install` – Expo dobierze wersje zgodne z SDK 58.

Terminal

```
npx create-expo-app@latest kino --template blank-typescript
cd kino
npx expo install expo-image expo-linear-gradient expo-font expo-splash-screen react-native-safe-area-context @expo-google-fonts/inter @expo/vector-icons
```

> **Informacja**
>
> Szablon `blank-typescript` zawiera już `expo`, `react`, `react-native`, `expo-status-bar` i plik `App.tsx`. Pliki z katalogu `assets/` (ikona, splash) zostawiamy z szablonu – własne ikony przygotujemy na wykładzie 9.

Pliki w tym etapie

Tabela pokazuje wszystkie pliki, które w tym etapie powstają (NOWY) lub się zmieniają (ZMIENIONY). Zmienione pliki są w instrukcji podane w CAŁOŚCI – wystarczy zastąpić ich zawartość.

Plik	Stan	Opis
`app.json`	NOWY	
`App.tsx`	NOWY	etap W02: jeden ekran, widoki przełączane stanem (nawigacja w W03)

Plik	Stan	Opis
src/components/AgeBadge.tsx	NOWY	oznaczenie kategorii wiekowej
src/components/AppText.tsx	NOWY	Text z typografią aplikacji
src/components/EmptyState.tsx	NOWY	komunikat dla pustej listy
src/components/GenreChips.tsx	NOWY	poziomy pasek filtrów gatunków
src/components/MovieCard.tsx	NOWY	kafelek filmu w siatce
src/components/MovieDetailsModal.tsx	NOWY	szczegóły filmu w oknie modalnym
src/components/PremiereCarousel.tsx	NOWY	poziomy karuzel premier
src/components/RatingStars.tsx	NOWY	ocena 0–10 jako 5 gwiazdek
src/components/ScreeningRow.tsx	NOWY	wiersz seansu w SectionList
src/components/SeatMap.tsx	NOWY	plan sali zbudowany wyłącznie z Flexboxa
src/components/SegmentedControl.tsx	NOWY	przełącznik widoków (do W03 zastępuje nawigację)
src/data/mockData.ts	NOWY	dane przykładowe (od W06 zastąpi je API)
src/data/types.ts	NOWY	modele domenowe aplikacji
src/screens/HallView.tsx	NOWY	plan sali dla wybranego seansu + pasek podsumowania
src/screens/RepertoireView.tsx	NOWY	repertuar: karuzel + filtry + responsywna siatka
src/screens/ScreeningsView.tsx	NOWY	seanse pogrupowane według dni (SectionList)
src/theme/theme.ts	NOWY	jedno źródło kolorów, odstępów i typografii
src/utils/format.ts	NOWY	formatowanie przez wbudowane Intl (bez dodatkowych bibliotek)
tsconfig.json	NOWY	

Krok po kroku

1. Konfiguracja projektu i alias importów

W `app.json` ustawiamy nazwę aplikacji, automatyczny motyw (`userInterfaceStyle: "automatic"` – bez tego tryb ciemny nie zadziała) i obsługę obu orientacji. W `tsconfig.json` dodajemy alias `@/*` → `./src/*`: zamiast `../../components/AppText` piszemy `@/components/AppText`. Metro w Expo rozumie aliasy z `tsconfig.json` bez dodatkowej konfiguracji.

Wszystkie nasze pliki trzymamy w katalogu `src/` (pliki z `app.json`, `App.tsx` i `tsconfig.json` zostają w głównym katalogu).

app.json

Nowy plik

```
{
  "expo": {
    "name": "Kino Pod Gwiazdami",
    "slug": "kino-pod-gwiazdami",
    "version": "1.0.0",
    "orientation": "default",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "automatic",
    "ios": { "supportsTablet": true },
    "android": {
      "adaptiveIcon": {
        "backgroundColor": "#E50914",
        "foregroundImage": "./assets/android-icon-foreground.png"
      }
    },
    "plugins": ["expo-font"]
  }
}
```

tsconfig.json

Nowy plik

```
{
  "extends": "expo/tsconfig.base",
  "compilerOptions": {
    "strict": true,
    "paths": { "@/*": ["./src/*"] }
  }
}
```

2. Model danych i dane przykładowe

Najpierw typy: `Movie`, `Hall`, `Screening`. Typ `Genre` to unia napisów – TypeScript podpowiada wartości i wyłapuje literówki. Dane przykładowe (filmy, sale, seanse na 3 dni) generujemy w `mockData.ts`; od wykładu 6 zastąpi je prawdziwe API.

src/data/types.ts

Nowy plik – modele domenowe aplikacji

```
// src/data/types.ts – modele domenowe aplikacji
export type AgeRating = "BO" | "7+" | "12+" | "16+" | "18+";
export type Genre =
  | "Akcja"
  | "Animacja"
```

```
  | "Dramat"
  | "Familijny"
  | "Horror"
  | "Komedia"
  | "Sci-Fi"
  | "Thriller";

export type Movie = {
  id: string;
  title: string;
  genres: Genre[];
  durationMin: number;
  ageRating: AgeRating;
  rating: number; // 0–10
  posterUrl: string;
  backdropUrl: string;
  description: string;
  director: string;
  releaseDate: string; // ISO, np. "2026-10-17"
  isPremiere: boolean;
};

export type HallType = "2D" | "3D" | "IMAX" | "4DX";

export type Hall = {
  id: string;
  name: string;
  rows: number;
  seatsPerRow: number;
  type: HallType;
  vipRows: number[]; // indeksy rzędów VIP (0 = pierwszy rząd)
};

export type ScreeningLanguage = "dubbing" | "napisy" | "oryginał";

export type Screening = {
  id: string;
  movieId: string;
  hallId: string;
  startsAt: string; // ISO
  language: ScreeningLanguage;
  price: number; // PLN
  takenSeats: string[]; // np. ["A5", "A6"]
};
```

src/data/mockData.ts

Nowy plik – dane przykładowe (od W06 zastąpi je API)

```
// src/data/mockData.ts – dane przykładowe (od W06 zastąpi je API)
import type { Genre, Hall, Movie, Screening } from "./types";

// Obrazki z picsum.photos – darmowe zdjęcia generowane po „seed”
const poster = (seed: string) => `https://picsum.photos/seed/${seed}/400/600`;
const backdrop = (seed: string) => `https://picsum.photos/seed/${seed}-bg/1200/675`;

export const GENRES: Genre[] = [
  "Akcja",
  "Animacja",
  "Dramat",
  "Familijny",
  "Horror",
  "Komedia",
  "Sci-Fi",
  "Thriller",
```

```
];

export const MOVIES: Movie[] = [
  {
    id: "m1",
    title: "Ostatni Seans",
    genres: ["Dramat", "Thriller"],
    durationMin: 128,
    ageRating: "16+",
    rating: 8.4,
    posterUrl: poster("seans"),
    backdropUrl: backdrop("seans"),
    description:
      "Kinooperator z małego miasteczka odkrywa, że ostatnia taśma z archiwum skrywa tajemnicę
sprzed 40 lat. Im dłużej ją ogląda, tym bardziej zaciera się granica między filmem a
rzeczywistością.",
    director: "Anna Wiśniewska",
    releaseDate: "2026-10-16",
    isPremiere: true,
  },
  {
    id: "m2",
    title: "Gwiezdny Kurier",
    genres: ["Sci-Fi", "Akcja"],
    durationMin: 142,
    ageRating: "12+",
    rating: 7.9,
    posterUrl: poster("kurier"),
    backdropUrl: backdrop("kurier"),
    description:
      "Kurierka międzyplanetarna dostaje przesyłkę, której nie wolno otworzyć. Pościg przez trzy
układy słoneczne zaczyna się w chwili, gdy paczka zaczyna mówić.",
    director: "Marek Zieliński",
    releaseDate: "2026-10-09",
    isPremiere: true,
  },
  {
    id: "m3",
    title: "Lis i Latarnia",
    genres: ["Animacja", "Familijny"],
    durationMin: 94,
    ageRating: "BO",
    rating: 8.1,
    posterUrl: poster("lis"),
    backdropUrl: backdrop("lis"),
    description:
      "Mały lis wyrusza nad morze, by naprawić latarnię, która od lat nie świeci. Ciepła
opowieść o przyjaźni i odwadze dla całej rodziny.",
    director: "Ola Nowak",
    releaseDate: "2026-09-25",
    isPremiere: false,
  },
  {
    id: "m4",
    title: "Piąte Piętro",
    genres: ["Horror"],
    durationMin: 101,
    ageRating: "18+",
    rating: 6.8,
    posterUrl: poster("pietro"),
    backdropUrl: backdrop("pietro"),
    description:
      "W kamienicy, która ma cztery piętra, winda zatrzymuje się na piątym. Lokatorzy
postanawiają sprawdzić, kto tam mieszka.",
    director: "Piotr Kamiński",
    releaseDate: "2026-10-02",
    isPremiere: false,
```

```
  },
  {
    id: "m5",
    title: "Wesele na Bis",
    genres: ["Komedia"],
    durationMin: 112,
    ageRating: "12+",
    rating: 7.2,
    posterUrl: poster("wesele"),
    backdropUrl: backdrop("wesele"),
    description:
      "Dwie rodziny, jedna sala weselna i podwójna rezerwacja na ten sam dzień. Nikt nie
zamierza ustąpić.",
    director: "Katarzyna Lewandowska",
    releaseDate: "2026-09-18",
    isPremiere: false,
  },
  {
    id: "m6",
    title: "Cisza nad Tatrami",
    genres: ["Dramat"],
    durationMin: 119,
    ageRating: "12+",
    rating: 8.7,
    posterUrl: poster("tatry"),
    backdropUrl: backdrop("tatry"),
    description:
      "Ratownik TOPR po latach wraca w góry, by odnaleźć zaginionego przyjaciela. Oparte na
prawdziwych wydarzeniach.",
    director: "Tomasz Wójcik",
    releaseDate: "2026-10-16",
    isPremiere: true,
  },
  {
    id: "m7",
    title: "Kod Zero",
    genres: ["Thriller", "Akcja"],
    durationMin: 133,
    ageRating: "16+",
    rating: 7.5,
    posterUrl: poster("kod"),
    backdropUrl: backdrop("kod"),
    description:
      "Programistka znajduje w systemie banku linijkę kodu, która za 48 godzin wyzeruje
wszystkie konta w kraju.",
    director: "Jakub Kowalczyk",
    releaseDate: "2026-09-11",
    isPremiere: false,
  },
  {
    id: "m8",
    title: "Robo-Kumple",
    genres: ["Animacja", "Komedia", "Familijny"],
    durationMin: 88,
    ageRating: "BO",
    rating: 7.0,
    posterUrl: poster("robo"),
    backdropUrl: backdrop("robo"),
    description:
      "Dwa roboty sprzątające z centrum handlowego postanawiają zostać detektywami.",
    director: "Ewa Dąbrowska",
    releaseDate: "2026-10-09",
    isPremiere: false,
  },
];

export const HALLS: Hall[] = [
```

```
  { id: "h1", name: "Sala 1", rows: 8, seatsPerRow: 12, type: "2D", vipRows: [6, 7] },
  { id: "h2", name: "Sala 2", rows: 10, seatsPerRow: 14, type: "3D", vipRows: [8, 9] },
  { id: "h3", name: "Sala IMAX", rows: 12, seatsPerRow: 16, type: "IMAX", vipRows: [10, 11] },
  { id: "h4", name: "Sala 4DX", rows: 6, seatsPerRow: 10, type: "4DX", vipRows: [] },
];

// Seanse generujemy względem „dzisiaj”, żeby dane zawsze były aktualne
function at(dayOffset: number, hour: number, minute = 0): string {
  const d = new Date();
  d.setDate(d.getDate() + dayOffset);
  d.setHours(hour, minute, 0, 0);
  return d.toISOString();
}

export const SCREENINGS: Screening[] = [
  { id: "s1", movieId: "m1", hallId: "h1", startsAt: at(0, 17, 30), language: "oryginał", price:
28, takenSeats: ["C5", "C6", "D7"] },
  { id: "s2", movieId: "m2", hallId: "h3", startsAt: at(0, 19, 0), language: "napisy", price:
39, takenSeats: ["F8", "F9", "G8", "G9"] },
  { id: "s3", movieId: "m3", hallId: "h2", startsAt: at(0, 15, 15), language: "dubbing", price:
24, takenSeats: [] },
  { id: "s4", movieId: "m6", hallId: "h1", startsAt: at(0, 20, 45), language: "oryginał", price:
28, takenSeats: ["H1", "H2"] },
  { id: "s5", movieId: "m4", hallId: "h4", startsAt: at(1, 22, 0), language: "napisy", price:
45, takenSeats: ["A1"] },
  { id: "s6", movieId: "m5", hallId: "h2", startsAt: at(1, 18, 30), language: "oryginał", price:
26, takenSeats: [] },
  { id: "s7", movieId: "m8", hallId: "h1", startsAt: at(1, 12, 0), language: "dubbing", price:
22, takenSeats: ["B3"] },
  { id: "s8", movieId: "m2", hallId: "h3", startsAt: at(2, 16, 0), language: "dubbing", price:
39, takenSeats: [] },
  { id: "s9", movieId: "m7", hallId: "h2", startsAt: at(2, 21, 15), language: "napisy", price:
28, takenSeats: ["E5"] },
];

export const findMovie = (id: string) => MOVIES.find((m) => m.id === id);
export const findHall = (id: string) => HALLS.find((h) => h.id === id);
```

3. Motyw, typografia i formatowanie

`theme.ts` to jedno źródło kolorów (jasny i ciemny wariant), odstępów i zaokrągleń. Hook `useAppTheme()` wybiera motyw na podstawie `useColorScheme()`. `AppText` to jedyny komponent tekstowy w aplikacji – zna warianty typografii (hero, title, subtitle, body, caption, label) i kolory motywu.

Daty, godziny i ceny formatujemy wbudowanym API `Intl` – bez bibliotek typu moment.js.

src/theme/theme.ts

Nowy plik – jedno źródło kolorów, odstępów i typografii

```
// src/theme/theme.ts – jedno źródło kolorów, odstępów i typografii
import { useColorScheme } from "react-native";

const brand = {
  primary: "#E50914", // czerwień kinowa
  primaryDark: "#B20710",
  gold: "#F5C518", // gwiazdki ocen
  success: "#2E9E5B",
  warning: "#F59E0B",
  danger: "#DC2626",
  info: "#2563EB",
};
```

```
export const lightTheme = {
  dark: false,
  colors: {
    ...brand,
    background: "#F6F6F8",
    surface: "#FFFFFF",
    surfaceAlt: "#EEEEF2",
    border: "#E2E2E8",
    text: "#14141A",
    textMuted: "#5E5E6B",
    onPrimary: "#FFFFFF",
    seatFree: "#D4D4DC",
    seatTaken: "#7A7A86",
    seatVip: "#C9A227",
  },
};

export const darkTheme: AppTheme = {
  dark: true,
  colors: {
    ...brand,
    background: "#0E0E12",
    surface: "#1A1A21",
    surfaceAlt: "#24242D",
    border: "#2E2E38",
    text: "#F4F4F6",
    textMuted: "#A3A3B2",
    onPrimary: "#FFFFFF",
    seatFree: "#3A3A46",
    seatTaken: "#6B6B78",
    seatVip: "#B8921E",
  },
};

export type AppTheme = typeof lightTheme;

// Skala odstępów – zamiast „magicznych liczb” w stylach
export const spacing = { xs: 4, sm: 8, md: 12, lg: 16, xl: 24, xxl: 32 } as const;
export const radius = { sm: 8, md: 12, lg: 16, pill: 999 } as const;

// Każda grubość kroju to osobna rodzina (na Androidzie fontWeight nie działa z własnymi
fontami)
export const fonts = {
  regular: "Inter_400Regular",
  medium: "Inter_500Medium",
  semibold: "Inter_600SemiBold",
  bold: "Inter_700Bold",
  black: "Inter_900Black",
} as const;

// Motyw zależny od ustawień systemu (jasny / ciemny)
export function useAppTheme(): AppTheme {
  const scheme = useColorScheme();
  return scheme === "dark" ? darkTheme : lightTheme;
}
```

src/components/AppText.tsx

Nowy plik – Text z typografią aplikacji

```
// src/components/AppText.tsx – Text z typografią aplikacji
import { StyleSheet, Text, type TextProps } from "react-native";
import { fonts, useAppTheme } from "@/theme/theme";

type Variant = "hero" | "title" | "subtitle" | "body" | "caption" | "label";
```

```
type AppTextProps = TextProps & {
  variant?: Variant;
  color?: string;
  muted?: boolean;
};

export function AppText({ variant = "body", color, muted, style, ...rest }: AppTextProps) {
  const theme = useAppTheme();
  const textColor = color ?? (muted ? theme.colors.textMuted : theme.colors.text);
  // ...rest przekazuje dalej wszystkie propsy Text (numberOfLines, onPress, selectable...)
  return <Text style={[styles[variant], { color: textColor }, style]} {...rest} />;
}

const styles = StyleSheet.create({
  hero: { fontFamily: fonts.black, fontSize: 30, lineHeight: 36 },
  title: { fontFamily: fonts.bold, fontSize: 22, lineHeight: 28 },
  subtitle: { fontFamily: fonts.semibold, fontSize: 17, lineHeight: 22 },
  body: { fontFamily: fonts.regular, fontSize: 15, lineHeight: 21 },
  caption: { fontFamily: fonts.medium, fontSize: 12, lineHeight: 16 },
  label: {
    fontFamily: fonts.semibold,
    fontSize: 12,
    letterSpacing: 0.6,
    textTransform: "uppercase",
  },
});
```

src/utils/format.ts

Nowy plik – formatowanie przez wbudowane Intl (bez dodatkowych bibliotek)

```
// src/utils/format.ts – formatowanie przez wbudowane Intl (bez dodatkowych bibliotek)
const timeFmt = new Intl.DateTimeFormat("pl-PL", { hour: "2-digit", minute: "2-digit" });
const dayFmt = new Intl.DateTimeFormat("pl-PL", { weekday: "long", day: "numeric", month: "long"
});
const priceFmt = new Intl.NumberFormat("pl-PL", { style: "currency", currency: "PLN" });

export const formatTime = (iso: string) => timeFmt.format(new Date(iso));
export const formatPrice = (value: number) => priceFmt.format(value);

export function formatDuration(minutes: number) {
  const h = Math.floor(minutes / 60);
  const m = minutes % 60;
  return h > 0 ? `${h} h ${m} min` : `${m} min`;
}

// „Dziś”, „Jutro” albo pełna data
export function formatDayLabel(iso: string) {
  const date = new Date(iso);
  const today = new Date();
  const diffDays = Math.round(
    (startOfDay(date).getTime() - startOfDay(today).getTime()) / 86_400_000,
  );
  if (diffDays === 0) return "Dziś";
  if (diffDays === 1) return "Jutro";
  return dayFmt.format(date);
}

export function startOfDay(date: Date) {
  const d = new Date(date);
  d.setHours(0, 0, 0, 0);
  return d;
}
```

```
// Oznaczenie miejsca: rząd 0 -> „A”, miejsce 4 -> „A5”
export const seatId = (row: number, seat: number) =>
  `${String.fromCharCode(65 + row)}${seat + 1}`;
```

4. Małe komponenty wielokrotnego użytku

Kilka prostych komponentów, z których zbudujemy widoki: plakietka wieku, gwiazdki oceny, poziomy pasek „chipsów” gatunków (poziomy `ScrollView`), komunikat pustej listy i przełącznik widoków `SegmentedControl` – generyczny komponent `SegmentedControl<T>`, który przyjmuje dowolną unię wartości.

src/components/AgeBadge.tsx

Nowy plik – oznaczenie kategorii wiekowej

```
// src/components/AgeBadge.tsx – oznaczenie kategorii wiekowej
import { StyleSheet, View } from "react-native";
import type { AgeRating } from "@/data/types";
import { AppText } from "./AppText";

const COLORS: Record<AgeRating, string> = {
  BO: "#2E9E5B",
  "7+": "#3B82F6",
  "12+": "#F59E0B",
  "16+": "#EA580C",
  "18+": "#DC2626",
};

export function AgeBadge({ rating }: { rating: AgeRating }) {
  return (
    <View
      style={[styles.badge, { backgroundColor: COLORS[rating] }]}
      accessibilityLabel={rating === "BO" ? "Bez ograniczeń wiekowych" : `Od ${rating} lat`}
    >
      <AppText variant="caption" color="#FFFFFF">
        {rating}
      </AppText>
    </View>
  );
}

const styles = StyleSheet.create({
  badge: {
    paddingHorizontal: 6,
    paddingVertical: 2,
    borderRadius: 6,
    alignSelf: "flex-start", // badge nie rozciąga się na całą szerokość rodzica
  },
});
```

src/components/RatingStars.tsx

Nowy plik – ocena 0–10 jako 5 gwiazdek

```
// src/components/RatingStars.tsx – ocena 0–10 jako 5 gwiazdek
import { StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

export function RatingStars({ rating, size = 14 }: { rating: number; size?: number }) {
  const theme = useAppTheme();
  const stars = rating / 2; // skala 0–5
```

```
  return (
    <View style={styles.row} accessible accessibilityLabel={`Ocena ${rating} na 10`}>
      {[0, 1, 2, 3, 4].map((i) => {
        const name = stars >= i + 1 ? "star" : stars >= i + 0.5 ? "star-half" : "star-outline";
        return <Ionicons key={i} name={name} size={size} color={theme.colors.gold} />;
      })}
      <AppText variant="caption" muted style={styles.value}>
        {rating.toFixed(1)}
      </AppText>
    </View>
  );
}

const styles = StyleSheet.create({
  row: { flexDirection: "row", alignItems: "center", gap: 2 },
  value: { marginLeft: 4 },
});
```

src/components/GenreChips.tsx

Nowy plik – poziomy pasek filtrów gatunków

```
// src/components/GenreChips.tsx – poziomy pasek filtrów gatunków
import { Pressable, ScrollView, StyleSheet } from "react-native";
import type { Genre } from "@/data/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props = {
  genres: Genre[];
  selected: Genre[];
  onToggle: (genre: Genre) => void;
};

export function GenreChips({ genres, selected, onToggle }: Props) {
  const theme = useAppTheme();
  return (
    <ScrollView
      horizontal
      showsHorizontalScrollIndicator={false}
      contentContainerStyle={styles.row}
      keyboardShouldPersistTaps="handled"
    >
      {genres.map((genre) => {
        const active = selected.includes(genre);
        return (
          <Pressable
            key={genre}
            onPress={() => onToggle(genre)}
            hitSlop={6} // większy obszar dotyku niż widoczny element
            style={[
              styles.chip,
              {
                backgroundColor: active ? theme.colors.primary : theme.colors.surface,
                borderColor: active ? theme.colors.primary : theme.colors.border,
              },
            ]}
            accessibilityRole="checkbox"
            accessibilityState={{ checked: active }}
          >
            <AppText variant="caption" color={active ? theme.colors.onPrimary :
theme.colors.text}>
              {genre}
            </AppText>
          </Pressable>
        );
```

```
      ))}
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  row: { paddingHorizontal: 16, gap: spacing.sm },
  chip: {
    paddingHorizontal: spacing.md,
    paddingVertical: spacing.sm,
    borderRadius: radius.pill,
    borderWidth: 1,
  },
});
```

src/components/EmptyState.tsx

Nowy plik – komunikat dla pustej listy

```
// src/components/EmptyState.tsx – komunikat dla pustej listy
import { StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props = {
  icon?: keyof typeof Ionicons.glyphMap;
  title: string;
  message?: string;
};

export function EmptyState({ icon = "film-outline", title, message }: Props) {
  const theme = useAppTheme();
  return (
    <View style={styles.container}>
      <Ionicons name={icon} size={48} color={theme.colors.textMuted} />
      <AppText variant="subtitle">{title}</AppText>
      {!!message && (
        <AppText muted style={styles.message}>
          {message}
        </AppText>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
  container: { alignItems: "center", padding: spacing.xxl, gap: spacing.sm },
  message: { textAlign: "center" },
});
```

src/components/SegmentedControl.tsx

Nowy plik – przełącznik widoków (do W03 zastępuje nawigację)

```
// src/components/SegmentedControl.tsx – przełącznik widoków (do W03 zastępuje nawigację)
import { Pressable, StyleSheet, View } from "react-native";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props<T extends string> = {
  options: { value: T; label: string }[];
  value: T;
  onChange: (value: T) => void;
};
```

```
// Komponent generyczny: T to unia dozwolonych wartości (np. "movies" | "screenings")
export function SegmentedControl<T extends string>({ options, value, onChange }: Props<T>) {
  const theme = useAppTheme();
  return (
    <View
      style={[styles.container, { backgroundColor: theme.colors.surfaceAlt }]}
      accessibilityRole="tablist"
    >
      {options.map((option) => {
        const active = option.value === value;
        return (
          <Pressable
            key={option.value}
            onPress={() => onChange(option.value)}
            style={[styles.segment, active && { backgroundColor: theme.colors.surface }]}
            accessibilityRole="tab"
            accessibilityState={{ selected: active }}
          >
            <AppText
              variant="caption"
              color={active ? theme.colors.text : theme.colors.textMuted}
            >
              {option.label}
            </AppText>
          </Pressable>
        );
      })}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flexDirection: "row",
    padding: 3,
    borderRadius: radius.md,
    marginHorizontal: spacing.lg,
  },
  segment: {
    flex: 1, // każdy segment zajmuje równą część szerokości
    alignItems: "center",
    paddingVertical: spacing.sm,
    borderRadius: radius.sm,
  },
});
```

5. Repertuar: karta filmu, karuzela premier i siatka FlatList

`MovieCard` to `Pressable` z obrazem `expo-image` (rozmyty podgląd blurhash, `contentFit="cover"`, `transition`). Na Androidzie dotyk pokazuje natywny efekt fali (`android_ripple`), na iOS przyciemnienie przez funkcję w `style` (`({ pressed }) => …`). Komponent jest opakowany w `memo`.

`PremiereCarousel` to poziomy `ScrollView` z `snapToInterval` – po puszczeniu palca lista „dociąga" do najbliższej karty. Szerokość karty liczymy z `useWindowDimensions`, więc karuzela działa także na tablecie i po obróceniu ekranu.

`RepertoireView` składa wszystko w `FlatList`: nagłówek listy (`ListHeaderComponent`) zawiera karuzelę, wyszukiwarkę `TextInput`, przełącznik `Switch` („tylko premiery") i chipsy gatunków. Liczba kolumn zależy od szerokości (2/3/4). Filtrowanie liczymy w `useMemo`, a funkcje przekazywane do kart stabilizujemy `useCallback`.

src/components/MovieCard.tsx

Nowy plik – kafelek filmu w siatce

```
// src/components/MovieCard.tsx – kafelek filmu w siatce
import { memo } from "react";
import { Platform, Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import type { Movie } from "@/data/types";
import { formatDuration } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";
import { AgeBadge } from "./AgeBadge";
import { RatingStars } from "./RatingStars";

// Prosty "blurhash" – rozmyta zaślepka widoczna, zanim obraz się pobierze
const BLURHASH = "L6PZfSi_.AyE_3t7t7R**0o#DgR4";

type Props = {
  movie: Movie;
  onPress: (movie: Movie) => void;
  onLongPress?: (movie: Movie) => void;
};

function MovieCardComponent({ movie, onPress, onLongPress }: Props) {
  const theme = useAppTheme();
  return (
    <Pressable
      onPress={() => onPress(movie)}
      onLongPress={() => onLongPress?.(movie)}
      delayLongPress={350}
      android_ripple={{ color: "rgba(0,0,0,0.15)", foreground: true }}
      // Funkcja w style dostaje stan { pressed } – efekt wciśnięcia na iOS
      style={({ pressed }) => [
        styles.card,
        { backgroundColor: theme.colors.surface, borderColor: theme.colors.border },
        pressed && Platform.OS === "ios" && styles.pressed,
      ]}
      accessibilityRole="button"
      accessibilityLabel={`${movie.title}, ${formatDuration(movie.durationMin)}`}
      accessibilityHint="Otwiera szczegóły filmu"
    >
      <Image
        source={movie.posterUrl}
        placeholder={{ blurhash: BLURHASH }}
        contentFit="cover"
        transition={250}
        style={styles.poster}
        accessibilityIgnoresInvertColors
      />
      <View style={styles.info}>
        <AppText variant="subtitle" numberOfLines={1}>
          {movie.title}
        </AppText>
        <View style={styles.metaRow}>
          <AgeBadge rating={movie.ageRating} />
          <AppText variant="caption" muted>
            {formatDuration(movie.durationMin)}
          </AppText>
        </View>
        <RatingStars rating={movie.rating} size={12} />
      </View>
      {movie.isPremiere && (
        <View style={[styles.ribbon, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="label" color={theme.colors.onPrimary}>
            Premiera
          </AppText>
```

```
        </View>
      )}
    </Pressable>
  );
}

// memo: karta przerenderuje się tylko, gdy zmienią się jej propsy
export const MovieCard = memo(MovieCardComponent);

const styles = StyleSheet.create({
  card: {
    flex: 1,
    borderRadius: radius.lg,
    borderWidth: StyleSheet.hairlineWidth, // najcieńsza linia na danym ekranie
    overflow: "hidden", // przycina obraz i efekt ripple do zaokrąglonych rogów
    boxShadow: "0 4px 14px rgba(0, 0, 0, 0.12)",
  },
  pressed: { opacity: 0.85, transform: [{ scale: 0.98 }] },
  poster: { width: "100%", aspectRatio: 2 / 3 },
  info: { padding: spacing.md, gap: 6 },
  metaRow: { flexDirection: "row", alignItems: "center", gap: spacing.sm },
  ribbon: {
    position: "absolute",
    top: spacing.sm,
    left: spacing.sm,
    paddingHorizontal: spacing.sm,
    paddingVertical: 3,
    borderRadius: radius.sm,
  },
});
```

src/components/PremiereCarousel.tsx

Nowy plik – poziomy karuzel premier

```
// src/components/PremiereCarousel.tsx – poziomy karuzel premier
import { useState } from "react";
import {
  Pressable,
  ScrollView,
  StyleSheet,
  View,
  useWindowDimensions,
  type NativeScrollEvent,
  type NativeSyntheticEvent,
} from "react-native";
import { Image } from "expo-image";
import { LinearGradient } from "expo-linear-gradient";
import type { Movie } from "@/data/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props = { movies: Movie[]; onPress: (movie: Movie) => void };

export function PremiereCarousel({ movies, onPress }: Props) {
  const theme = useAppTheme();
  const { width } = useWindowDimensions();
  const cardWidth = Math.min(width - spacing.lg * 2, 520); // na tablecie karta nie jest za
szeroka
  const [index, setIndex] = useState(0);

  // onScroll + scrollEventThrottle: odczyt pozycji, by zaznaczyć aktywną kropkę
  const onScroll = (e: NativeSyntheticEvent<NativeScrollEvent>) => {
    const x = e.nativeEvent.contentOffset.x;
    setIndex(Math.round(x / (cardWidth + spacing.md)));
  };
```

```
  return (
    <View>
      <ScrollView
        horizontal
        showsHorizontalScrollIndicator={false}
        snapToInterval={cardWidth + spacing.md} // „przyklejanie” do kolejnej karty
        decelerationRate="fast"
        contentContainerStyle={styles.content}
        onScroll={onScroll}
        scrollEventThrottle={16} // ~60 zdarzeń na sekundę
      >
        {movies.map((movie) => (
          <Pressable
            key={movie.id}
            onPress={() => onPress(movie)}
            style={[styles.card, { width: cardWidth }]}
            accessibilityRole="button"
            accessibilityLabel={`Premiera: ${movie.title}`}
          >
            <Image source={movie.backdropUrl} contentFit="cover" style={StyleSheet.absoluteFill}
/>
            {/* Gradient przyciemnia dół zdjęcia, żeby tekst był czytelny */}
            <LinearGradient
              colors={["transparent", "rgba(0,0,0,0.85)"]}
              style={StyleSheet.absoluteFill}
            />
            <View style={styles.caption}>
              <AppText variant="label" color={theme.colors.gold}>
                Premiera
              </AppText>
              <AppText variant="title" color="#FFFFFF" numberOfLines={1}>
                {movie.title}
              </AppText>
              <AppText variant="caption" color="#E5E5EA">
                {movie.genres.join(" · ")}
              </AppText>
            </View>
          </Pressable>
        ))}
      </ScrollView>

      <View style={styles.dots} accessibilityElementsHidden importantForAccessibility="no">
        {movies.map((m, i) => (
          <View
            key={m.id}
            style={[
              styles.dot,
              { backgroundColor: i === index ? theme.colors.primary : theme.colors.border },
              i === index && styles.dotActive,
            ]}
          />
        ))}
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  content: { paddingHorizontal: spacing.lg, gap: spacing.md },
  card: {
    aspectRatio: 16 / 9,
    borderRadius: radius.lg,
    overflow: "hidden",
    justifyContent: "flex-end",
  },
  caption: { padding: spacing.lg, gap: 2 },
```

```
  dots: { flexDirection: "row", justifyContent: "center", gap: 6, marginTop: spacing.sm },
  dot: { width: 6, height: 6, borderRadius: 3 },
  dotActive: { width: 18 },
});
```

src/screens/RepertoireView.tsx

Nowy plik – repertuar: karuzel + filtry + responsywna siatka

```
// src/screens/RepertoireView.tsx – repertuar: karuzel + filtry + responsywna siatka
import { useCallback, useMemo, useState } from "react";
import {
  Alert,
  FlatList,
  StyleSheet,
  Switch,
  TextInput,
  View,
  useWindowDimensions,
} from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { GENRES, MOVIES } from "@/data/mockData";
import type { Genre, Movie } from "@/data/types";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { GenreChips } from "@/components/GenreChips";
import { MovieCard } from "@/components/MovieCard";
import { PremiereCarousel } from "@/components/PremiereCarousel";
import { fonts, radius, spacing, useAppTheme } from "@/theme/theme";

type Props = { onOpenMovie: (movie: Movie) => void };

export function RepertoireView({ onOpenMovie }: Props) {
  const theme = useAppTheme();
  const { width } = useWindowDimensions();
  const [query, setQuery] = useState("");
  const [genres, setGenres] = useState<Genre[]>([]);
  const [onlyPremieres, setOnlyPremieres] = useState(false);
  const [refreshing, setRefreshing] = useState(false);

  // Liczba kolumn zależy od szerokości okna (telefon / obrót / tablet)
  const numColumns = width >= 900 ? 4 : width >= 600 ? 3 : 2;

  // Filtrowanie liczymy tylko, gdy zmienią się dane wejściowe
  const filtered = useMemo(
    () =>
      MOVIES.filter((m) => m.title.toLowerCase().includes(query.trim().toLowerCase()))
        .filter((m) => genres.length === 0 || m.genres.some((g) => genres.includes(g)))
        .filter((m) => !onlyPremieres || m.isPremiere),
    [query, genres, onlyPremieres],
  );

  const toggleGenre = (genre: Genre) =>
    setGenres((prev) =>
      prev.includes(genre) ? prev.filter((g) => g !== genre) : [...prev, genre],
    );

  // Symulacja „pociągnij, aby odświeżyć” (od W06 – prawdziwe pobranie z API)
  const onRefresh = () => {
    setRefreshing(true);
    setTimeout(() => setRefreshing(false), 1200);
  };

  // useCallback: stała referencja funkcji – memo w MovieCard ma sens
  const onLongPress = useCallback((movie: Movie) => {
```

```
    Alert.alert(movie.title, "Co chcesz zrobić?", [
      { text: "Pokaż szczegóły", onPress: () => onOpenMovie(movie) },
      { text: "Anuluj", style: "cancel" },
    ]);
  }, [onOpenMovie]);

  const header = (
    <View style={styles.header}>
      <PremiereCarousel movies={MOVIES.filter((m) => m.isPremiere)} onPress={onOpenMovie} />

      <View style={[styles.search, { backgroundColor: theme.colors.surface }]}>
        <Ionicons name="search" size={18} color={theme.colors.textMuted} />
        <TextInput
          value={query}
          onChangeText={setQuery}
          placeholder="Szukaj filmu..."
          placeholderTextColor={theme.colors.textMuted}
          style={[styles.searchInput, { color: theme.colors.text }]}
          returnKeyType="search"
          clearButtonMode="while-editing" // iOS: przycisk czyszczenia
          autoCorrect={false}
        />
      </View>

      <GenreChips genres={GENRES} selected={genres} onToggle={toggleGenre} />

      <View style={styles.switchRow}>
        <AppText>Tylko premiery</AppText>
        <Switch
          value={onlyPremieres}
          onValueChange={setOnlyPremieres}
          trackColor={{ true: theme.colors.primary, false: theme.colors.border }}
          thumbColor="#FFFFFF"
        />
      </View>
      <AppText variant="label" muted style={styles.count}>
        Filmy: {filtered.length}
      </AppText>
    </View>
  );

  return (
    <FlatList
      // Zmiana numColumns „w locie” wymaga nowego klucza – inaczej błąd
      key={numColumns}
      data={filtered}
      keyExtractor={(item) => item.id}
      numColumns={numColumns}
      renderItem={({ item }) => (
        <View style={[styles.cell, { maxWidth: `${100 / numColumns}%` }]}>
          <MovieCard movie={item} onPress={onOpenMovie} onLongPress={onLongPress} />
        </View>
      )}
      ListHeaderComponent={header}
      ListEmptyComponent={
        <EmptyState title="Brak filmów" message="Zmień kryteria wyszukiwania lub filtry." />
      }
      columnWrapperStyle={styles.columns}
      contentContainerStyle={styles.content}
      refreshing={refreshing}
      onRefresh={onRefresh}
      keyboardDismissMode="on-drag" // przewinięcie listy chowa klawiaturę
      initialNumToRender={6}
    />
  );
}
```

```
const styles = StyleSheet.create({
  header: { gap: spacing.lg, paddingTop: spacing.md, marginBottom: spacing.md },
  search: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.sm,
    marginHorizontal: spacing.lg,
    paddingHorizontal: spacing.md,
    borderRadius: radius.md,
    height: 44,
  },
  searchInput: { flex: 1, fontFamily: fonts.regular, fontSize: 15 },
  switchRow: {
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "space-between",
    marginHorizontal: spacing.lg,
  },
  count: { marginHorizontal: spacing.lg },
  content: { paddingBottom: spacing.xxl },
  columns: { paddingHorizontal: spacing.lg - spacing.sm / 2 },
  cell: { flex: 1, padding: spacing.sm / 2, paddingBottom: spacing.md },
});
```

6. Seanse: SectionList z przyklejanymi nagłówkami

Seanse grupujemy według dnia – `SectionList` dostaje tablicę sekcji `{ title, data }`. Nagłówek dnia „przykleja się" u góry podczas przewijania (`stickySectionHeadersEnabled`). `ScreeningRow` to jeden wiersz z godziną, filmem, salą i ceną.

src/components/ScreeningRow.tsx

Nowy plik – wiersz seansu w SectionList

```
// src/components/ScreeningRow.tsx – wiersz seansu w SectionList
import { Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { Ionicons } from "@expo/vector-icons";
import type { Hall, Movie, Screening } from "@/data/types";
import { formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props = {
  screening: Screening;
  movie: Movie;
  hall: Hall;
  onPress: (screening: Screening) => void;
};

export function ScreeningRow({ screening, movie, hall, onPress }: Props) {
  const theme = useAppTheme();
  const capacity = hall.rows * hall.seatsPerRow;
  const free = capacity - screening.takenSeats.length;

  return (
    <Pressable
      onPress={() => onPress(screening)}
      android_ripple={{ color: theme.colors.border }}
      style={[styles.row, { backgroundColor: theme.colors.surface }]}
      accessibilityRole="button"
      accessibilityLabel={`${movie.title}, godzina ${formatTime(screening.startsAt)},
${hall.name}`}
    >
```

```
      <View style={[styles.time, { backgroundColor: theme.colors.surfaceAlt }]}>
        <AppText variant="subtitle">{formatTime(screening.startsAt)}</AppText>
        <AppText variant="caption" muted>
          {hall.type}
        </AppText>
      </View>
      <Image source={movie.posterUrl} style={styles.thumb} contentFit="cover" />
      {/* flex: 1 – środkowa kolumna zajmuje całe wolne miejsce */}
      <View style={styles.middle}>
        <AppText variant="subtitle" numberOfLines={1}>
          {movie.title}
        </AppText>
        <AppText variant="caption" muted numberOfLines={1}>
          {hall.name} · {screening.language} · {formatPrice(screening.price)}
        </AppText>
        <AppText
          variant="caption"
          color={free < 10 ? theme.colors.warning : theme.colors.success}
        >
          Wolne miejsca: {free}
        </AppText>
      </View>
      <Ionicons name="chevron-forward" size={18} color={theme.colors.textMuted} />
    </Pressable>
  );
}

const styles = StyleSheet.create({
  row: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.md,
    marginHorizontal: spacing.lg,
    borderRadius: radius.md,
    overflow: "hidden",
  },
  time: {
    width: 64,
    alignItems: "center",
    paddingVertical: spacing.sm,
    borderRadius: radius.sm,
  },
  thumb: { width: 40, height: 60, borderRadius: 6 },
  middle: { flex: 1, gap: 2 },
});
```

src/screens/ScreeningsView.tsx

Nowy plik – seanse pogrupowane według dni (SectionList)

```
// src/screens/ScreeningsView.tsx – seanse pogrupowane według dni (SectionList)
import { useMemo } from "react";
import { SectionList, StyleSheet, View } from "react-native";
import { SCREENINGS, findHall, findMovie } from "@/data/mockData";
import type { Screening } from "@/data/types";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { ScreeningRow } from "@/components/ScreeningRow";
import { formatDayLabel, startOfDay } from "@/utils/format";
import { spacing, useAppTheme } from "@/theme/theme";

type Section = { title: string; data: Screening[] };

type Props = { onSelectScreening: (screening: Screening) => void };
```

```
export function ScreeningsView({ onSelectScreening }: Props) {
  const theme = useAppTheme();

  // Grupujemy seanse po dniu i sortujemy po godzinie
  const sections = useMemo<Section[]>(() => {
    const byDay = new Map<number, Screening[]>();
    for (const s of [...SCREENINGS].sort((a, b) => a.startsAt.localeCompare(b.startsAt))) {
      const key = startOfDay(new Date(s.startsAt)).getTime();
      byDay.set(key, [...(byDay.get(key) ?? []), s]);
    }
    return [...byDay.entries()].map(([, data]) => ({
      title: formatDayLabel(data[0].startsAt),
      data,
    }));
  }, []);

  return (
    <SectionList
      sections={sections}
      keyExtractor={(item) => item.id}
      renderItem={({ item }) => {
        const movie = findMovie(item.movieId);
        const hall = findHall(item.hallId);
        if (!movie || !hall) return null;
        return (
          <ScreeningRow screening={item} movie={movie} hall={hall} onPress={onSelectScreening}
/>
        );
      }}
      renderSectionHeader={({ section }) => (
        <View style={[styles.header, { backgroundColor: theme.colors.background }]}>
          <AppText variant="title">{section.title}</AppText>
          <AppText variant="caption" muted>
            Seanse: {section.data.length}
          </AppText>
        </View>
      )}
      stickySectionHeadersEnabled // nagłówek dnia „przykleja się” podczas przewijania
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      SectionSeparatorComponent={() => <View style={styles.sectionGap} />}
      ListEmptyComponent={<EmptyState icon="calendar-outline" title="Brak seansów" />}
      ListFooterComponent={
        <AppText variant="caption" muted style={styles.footer}>
          Repertuar może ulec zmianie.
        </AppText>
      }
      contentContainerStyle={styles.content}
    />
  );
}

const styles = StyleSheet.create({
  header: {
    flexDirection: "row",
    alignItems: "baseline",
    justifyContent: "space-between",
    paddingHorizontal: spacing.lg,
    paddingVertical: spacing.sm,
  },
  separator: { height: spacing.sm },
  sectionGap: { height: spacing.sm },
  footer: { textAlign: "center", marginTop: spacing.xl },
  content: { paddingBottom: spacing.xxl },
});
```

7. Plan sali z samego Flexboxa

Plan sali to dobre ćwiczenie z Flexboxa: rzędy (`flexDirection: "row"`), przejście w środku sali (pusty `View` o stałej szerokości), oznaczenia rzędów i rzędy VIP. Pojedynczy fotel to zmemoizowany `SeatCell` – po kliknięciu jednego fotela nie przerysowujemy setek pozostałych.

`HallView` pokazuje plan dla wybranego seansu i pasek podsumowania przyklejony do dołu. Pasek uwzględnia „bezpieczny obszar" (`useSafeAreaInsets().bottom`) – na telefonach z gestami systemowymi nie wchodzi pod pasek gestów.

src/components/SeatMap.tsx

Nowy plik – plan sali zbudowany wyłącznie z Flexboxa

```
// src/components/SeatMap.tsx – plan sali zbudowany wyłącznie z Flexboxa
import { Pressable, StyleSheet, View } from "react-native";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import type { Hall } from "@/data/types";
import { seatId } from "@/utils/format";
import { spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";

type Props = {
  hall: Hall;
  taken: string[];
  selected: string[];
  onToggle: (seat: string) => void;
  seatSize?: number;
};

export function SeatMap({ hall, taken, selected, onToggle, seatSize = 22 }: Props) {
  const theme = useAppTheme();
  const aisleAfter = Math.floor(hall.seatsPerRow / 2) - 1; // przejście na środku rzędu

  return (
    <View style={styles.container}>
      {/* Ekran: zaokrąglona belka nad salą */}
      <View style={[styles.screen, { borderColor: theme.colors.primary }]} />
      <AppText variant="label" muted style={styles.screenLabel}>
        Ekran
      </AppText>

      {Array.from({ length: hall.rows }, (_, row) => (
        <View key={row} style={styles.row}>
          <AppText variant="caption" muted style={styles.rowLabel}>
            {String.fromCharCode(65 + row)}
          </AppText>
          {Array.from({ length: hall.seatsPerRow }, (_, seat) => {
            const id = seatId(row, seat);
            const isTaken = taken.includes(id);
            const isSelected = selected.includes(id);
            const isVip = hall.vipRows.includes(row);
            const color = isTaken
              ? theme.colors.seatTaken
              : isSelected
                ? theme.colors.primary
                : isVip
                  ? theme.colors.seatVip
                  : theme.colors.seatFree;
            return (
              <Pressable
                key={id}
                disabled={isTaken}
                onPress={() => onToggle(id)}
                hitSlop={2}
```

```
                style={seat === aisleAfter ? styles.aisle : undefined}
                accessibilityRole="checkbox"
                accessibilityLabel={`Miejsce ${id}${isVip ? ", VIP" : ""}`}
                accessibilityState={{ checked: isSelected, disabled: isTaken }}
              >
                <MaterialCommunityIcons
                  name={isTaken ? "seat" : isSelected ? "seat" : "seat-outline"}
                  size={seatSize}
                  color={color}
                />
              </Pressable>
            );
          })}
        </View>
      ))}

      <View style={styles.legend}>
        <Legend color={theme.colors.seatFree} label="Wolne" />
        <Legend color={theme.colors.seatVip} label="VIP" />
        <Legend color={theme.colors.primary} label="Wybrane" />
        <Legend color={theme.colors.seatTaken} label="Zajęte" />
      </View>
    </View>
  );
}

function Legend({ color, label }: { color: string; label: string }) {
  return (
    <View style={styles.legendItem}>
      <View style={[styles.legendDot, { backgroundColor: color }]} />
      <AppText variant="caption" muted>
        {label}
      </AppText>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { alignItems: "center", gap: 4 },
  screen: {
    width: "80%",
    height: 18,
    borderTopWidth: 4,
    borderTopLeftRadius: 120, // „wygięty” ekran
    borderTopRightRadius: 120,
  },
  screenLabel: { marginBottom: spacing.md },
  row: { flexDirection: "row", alignItems: "center", gap: 2 },
  rowLabel: { width: 18, textAlign: "center" },
  aisle: { marginRight: spacing.md },
  legend: {
    flexDirection: "row",
    flexWrap: "wrap", // na wąskim ekranie legenda przejdzie do nowej linii
    justifyContent: "center",
    gap: spacing.md,
    marginTop: spacing.lg,
  },
  legendItem: { flexDirection: "row", alignItems: "center", gap: 6 },
  legendDot: { width: 12, height: 12, borderRadius: 3 },
});
```

src/screens/HallView.tsx

Nowy plik – plan sali dla wybranego seansu + pasek podsumowania

```
// src/screens/HallView.tsx – plan sali dla wybranego seansu + pasek podsumowania
```

```
import { useState } from "react";
import { Pressable, ScrollView, StyleSheet, View, useWindowDimensions } from "react-native";
import { useSafeAreaInsets } from "react-native-safe-area-context";
import { findHall, findMovie } from "@/data/mockData";
import type { Screening } from "@/data/types";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { SeatMap } from "@/components/SeatMap";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type Props = { screening: Screening | null };

export function HallView({ screening }: Props) {
  const theme = useAppTheme();
  const insets = useSafeAreaInsets();
  const { width } = useWindowDimensions();
  const [selected, setSelected] = useState<string[]>([]);

  if (!screening) {
    return (
      <EmptyState
        icon="grid-outline"
        title="Wybierz seans"
        message="Przejdź do zakładki „Seanse” i dotknij wybranego seansu."
      />
    );
  }

  const hall = findHall(screening.hallId);
  const movie = findMovie(screening.movieId);
  if (!hall || !movie) return null;

  // Rozmiar fotela dopasowany do szerokości ekranu i liczby miejsc w rzędzie
  const seatSize = Math.min(28, Math.floor((width - 80) / (hall.seatsPerRow + 1)));

  const toggleSeat = (seat: string) =>
    setSelected((prev) =>
      prev.includes(seat) ? prev.filter((s) => s !== seat) : [...prev, seat],
    );

  return (
    <View style={styles.flex}>
      <ScrollView contentContainerStyle={styles.content}>
        <AppText variant="title">{movie.title}</AppText>
        <AppText muted>
          {formatDayLabel(screening.startsAt)}, {formatTime(screening.startsAt)} · {hall.name} (
          {hall.type})
        </AppText>
        {/* Poziomy ScrollView – szeroka sala zmieści się także na małym telefonie */}
        <ScrollView horizontal contentContainerStyle={styles.mapScroll}>
          <SeatMap
            hall={hall}
            taken={screening.takenSeats}
            selected={selected}
            onToggle={toggleSeat}
            seatSize={seatSize}
          />
        </ScrollView>
      </ScrollView>

      {/* Pasek przyklejony do dołu, z odstępem nad paskiem gestów systemu */}
      <View
        style={[
          styles.summary,
```

```
          {
            backgroundColor: theme.colors.surface,
            borderTopColor: theme.colors.border,
            paddingBottom: insets.bottom + spacing.md,
          },
        ]}
      >
        <View style={styles.flex}>
          <AppText variant="caption" muted>
            {selected.length > 0 ? `Miejsca: ${selected.join(", ")}` : "Nie wybrano miejsc"}
          </AppText>
          <AppText variant="subtitle">{formatPrice(selected.length * screening.price)}</AppText>
        </View>
        <Pressable
          disabled={selected.length === 0}
          onPress={() => setSelected([])}
          style={({ pressed }) => [
            styles.button,
            { backgroundColor: theme.colors.primary },
            (pressed || selected.length === 0) && styles.buttonDim,
          ]}
          accessibilityRole="button"
        >
          <AppText variant="subtitle" color={theme.colors.onPrimary}>
            Wyczyść
          </AppText>
        </Pressable>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.sm },
  mapScroll: { paddingVertical: spacing.xl, minWidth: "100%", justifyContent: "center" },
  summary: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    paddingHorizontal: spacing.lg,
    paddingTop: spacing.md,
    borderTopWidth: StyleSheet.hairlineWidth,
  },
  button: {
    paddingHorizontal: spacing.xl,
    paddingVertical: spacing.md,
    borderRadius: radius.md,
  },
  buttonDim: { opacity: 0.5 },
});
```

8. Szczegóły filmu w oknie Modal

`Modal` z React Native wyświetla treść nad całą aplikacją. Na iOS `presentationStyle="pageSheet"` daje natywny arkusz zamykany gestem w dół. `onRequestClose` jest **wymagane** na Androidzie – obsługuje systemowy przycisk/gest „Wstecz”.

src/components/MovieDetailsModal.tsx

Nowy plik – szczegóły filmu w oknie modalnym

```
// src/components/MovieDetailsModal.tsx – szczegóły filmu w oknie modalnym
import { useState } from "react";
import { Modal, Pressable, ScrollView, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
```

```
import { Ionicons } from "@expo/vector-icons";
import { useSafeAreaInsets } from "react-native-safe-area-context";
import type { Movie } from "@/data/types";
import { formatDuration } from "@/utils/format";
import { fonts, radius, spacing, useAppTheme } from "@/theme/theme";
import { AppText } from "./AppText";
import { AgeBadge } from "./AgeBadge";
import { RatingStars } from "./RatingStars";

type Props = { movie: Movie | null; onClose: () => void };

export function MovieDetailsModal({ movie, onClose }: Props) {
  const theme = useAppTheme();
  const insets = useSafeAreaInsets();
  const [expanded, setExpanded] = useState(false);

  return (
    <Modal
      visible={movie !== null}
      animationType="slide"
      presentationStyle="pageSheet" // iOS: karta nasunięta na ekran; Android: pełny ekran
      onRequestClose={onClose} // Android: systemowy przycisk „wstecz”
      onDismiss={() => setExpanded(false)}
    >
      {movie && (
        <View style={[styles.flex, { backgroundColor: theme.colors.background }]}>
          <ScrollView contentContainerStyle={{ paddingBottom: insets.bottom + spacing.xl }}>
            <Image source={movie.backdropUrl} style={styles.backdrop} contentFit="cover" />
            <View style={styles.body}>
              <AppText variant="hero" selectable>
                {movie.title}
              </AppText>
              <View style={styles.meta}>
                <AgeBadge rating={movie.ageRating} />
                <AppText muted>{formatDuration(movie.durationMin)}</AppText>
                <RatingStars rating={movie.rating} />
              </View>
              <AppText muted>Reżyseria: {movie.director}</AppText>
              {/* Zagnieżdżony Text dziedziczy styl rodzica i może go nadpisać */}
              <AppText numberOfLines={expanded ? undefined : 3}>
                <AppText style={styles.bold}>Opis: </AppText>
                {movie.description}
              </AppText>
              <Pressable onPress={() => setExpanded((v) => !v)} hitSlop={8}>
                <AppText color={theme.colors.primary}>
                  {expanded ? "Zwiń" : "Czytaj więcej"}
                </AppText>
              </Pressable>
            </View>
          </ScrollView>

          <Pressable
            onPress={onClose}
            style={[styles.close, { backgroundColor: "rgba(0,0,0,0.55)" }]}
            accessibilityRole="button"
            accessibilityLabel="Zamknij"
            hitSlop={10}
          >
            <Ionicons name="close" size={22} color="#FFFFFF" />
          </Pressable>
        </View>
      )}
    </Modal>
  );
}
```

```
const styles = StyleSheet.create({
  flex: { flex: 1 },
  backdrop: { width: "100%", aspectRatio: 16 / 9 },
  body: { padding: spacing.lg, gap: spacing.md },
  meta: { flexDirection: "row", alignItems: "center", gap: spacing.md, flexWrap: "wrap" },
  // Własny font: pogrubienie przez inną rodzinę, nie fontWeight (Android)
  bold: { fontFamily: fonts.bold },
  close: {
    position: "absolute",
    top: spacing.lg,
    right: spacing.lg,
    width: 36,
    height: 36,
    borderRadius: radius.pill,
    alignItems: "center",
    justifyContent: "center",
  },
});
```

9. App.tsx – fonty, splash i bezpieczny obszar

`SplashScreen.preventAutoHideAsync()` wywołujemy na poziomie modułu (przed pierwszym renderem), a `hideAsync()` – gdy fonty się załadują. Dopóki fonty się ładują, zwracamy `null` (widoczny jest splash). Całość opakowujemy w `SafeAreaProvider`, a ekran w `SafeAreaView` z wybranymi krawędziami.

App.tsx

Nowy plik – etap W02: jeden ekran, widoki przełączane stanem (nawigacja w W03)

```
// App.tsx – etap W02: jeden ekran, widoki przełączane stanem (nawigacja w W03)
import { useEffect, useState } from "react";
import { StyleSheet, View } from "react-native";
import { StatusBar } from "expo-status-bar";
import * as SplashScreen from "expo-splash-screen";
import { SafeAreaProvider, SafeAreaView } from "react-native-safe-area-context";
import {
  Inter_400Regular,
  Inter_500Medium,
  Inter_600SemiBold,
  Inter_700Bold,
  Inter_900Black,
  useFonts,
} from "@expo-google-fonts/inter";
import type { Movie, Screening } from "@/data/types";
import { AppText } from "@/components/AppText";
import { MovieDetailsModal } from "@/components/MovieDetailsModal";
import { SegmentedControl } from "@/components/SegmentedControl";
import { HallView } from "@/screens/HallView";
import { RepertoireView } from "@/screens/RepertoireView";
import { ScreeningsView } from "@/screens/ScreeningsView";
import { spacing, useAppTheme } from "@/theme/theme";

// Ekran startowy (splash) zostaje, dopóki nie załadujemy fontów
SplashScreen.preventAutoHideAsync();

type Tab = "movies" | "screenings" | "hall";

export default function App() {
  const [fontsLoaded, fontError] = useFonts({
    Inter_400Regular,
    Inter_500Medium,
    Inter_600SemiBold,
    Inter_700Bold,
    Inter_900Black,
```

```
  });

  useEffect(() => {
    if (fontsLoaded || fontError) SplashScreen.hideAsync();
  }, [fontsLoaded, fontError]);

  if (!fontsLoaded && !fontError) return null;

  return (
    <SafeAreaProvider>
      <Main />
    </SafeAreaProvider>
  );
}

function Main() {
  const theme = useAppTheme();
  const [tab, setTab] = useState<Tab>("movies");
  const [movie, setMovie] = useState<Movie | null>(null);
  const [screening, setScreening] = useState<Screening | null>(null);

  const openScreening = (s: Screening) => {
    setScreening(s);
    setTab("hall");
  };

  return (
    // Dół obsługuje pasek podsumowania w HallView – tu tylko góra i boki
    <SafeAreaView
      style={[styles.flex, { backgroundColor: theme.colors.background }]}
      edges={["top", "left", "right"]}
    >
      {/* „auto": ciemne ikony w jasnym motywie, jasne w ciemnym */}
      <StatusBar style="auto" />
      <View style={styles.header}>
        <AppText variant="label" color={theme.colors.primary}>
          Kino Pod Gwiazdami
        </AppText>
        <AppText variant="hero">Co dziś oglądamy?</AppText>
      </View>
      <SegmentedControl<Tab>
        options={[
          { value: "movies", label: "Repertuar" },
          { value: "screenings", label: "Seanse" },
          { value: "hall", label: "Sala" },
        ]}
        value={tab}
        onChange={setTab}
      />
      <View style={styles.flex}>
        {tab === "movies" && <RepertoireView onOpenMovie={setMovie} />}
        {tab === "screenings" && <ScreeningsView onSelectScreening={openScreening} />}
        {tab === "hall" && <HallView key={screening?.id} screening={screening} />}
      </View>
      <MovieDetailsModal movie={movie} onClose={() => setMovie(null)} />
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  header: { paddingHorizontal: spacing.lg, paddingVertical: spacing.md, gap: 2 },
});
```

Uruchomienie i sprawdzenie

Terminal

```
npx expo start
```

Zeskanuj kod QR aplikacją Expo Go (Android) lub aparatem (iOS). Klawisz `a` otwiera emulator Androida, `i` symulator iOS, `w` – wersję w przeglądarce.

- ☐ przewiń karuzelę premier – karty „dociągają” do krawędzi,
- ☐ wpisz fragment tytułu i włącz „tylko premiery” – siatka filtruje się na bieżąco,
- ☐ obróć telefon / zmień szerokość okna przeglądarki – zmienia się liczba kolumn,
- ☐ w „Seansach” przewiń listę – nagłówek dnia zostaje u góry; wybierz seans – otworzy się plan sali,
- ☐ zaznacz kilka miejsc – pasek podsumowania liczy cenę; miejsca zajęte nie reagują,
- ☐ przełącz telefon w tryb ciemny – aplikacja zmienia kolory bez restartu.

Na co uważać

Cały tekst musi być w <Text>

W React Native napis bezpośrednio w `<View>` powoduje błąd „Text strings must be rendered within a <Text> component”. Uważaj też na `{count && <X/>}` – gdy `count` to `0`, renderuje się goły napis „0”. Używaj `{count > 0 && …}`.

Zmiana numColumns w FlatList

`numColumns` nie może zmienić się „w locie” – FlatList zgłosi błąd. Dlatego lista dostaje `key={numColumns}`: zmiana liczby kolumn tworzy nową listę.

Fonty a fontWeight

Przy własnych fontach `fontWeight: "bold"` na Androidzie zwykle nie działa – każda grubość to osobna rodzina (`Inter_700Bold`). Dlatego warianty w `AppText` ustawiają `fontFamily`, a nie `fontWeight`.

SafeAreaView z react-native jest przestarzały

Używaj `SafeAreaView` i `useSafeAreaInsets` z `react-native-safe-area-context` (działa na Androidzie i w trybie edge-to-edge, który od Androida 15 jest domyślny).

Cienie: boxShadow zamiast shadow*/elevation

Od nowej architektury RN obsługuje styl `boxShadow` (jak w CSS) na obu platformach. Stare `shadowColor/shadowOffset` działały tylko na iOS, a `elevation` tylko na Androidzie.

Po zmianie aliasów w tsconfig.json

Zrestartuj Metro z czyszczeniem cache: `npx expo start -c`. Inaczej import `@/…` może nie zostać rozpoznany.

Lista kontrolna

- ☐ projekt uruchamia się bez ostrzeżeń w Expo Go,
- ☐ wszystkie importy używają aliasu `@/…`,
- ☐ siatka ma 2 kolumny na telefonie i więcej na szerszym ekranie,
- ☐ Modal zamyka się przyciskiem „Wstecz” na Androidzie,
- ☐ aplikacja wygląda poprawnie w trybie jasnym i ciemnym.

Zadania dodatkowe

1. Dodaj do `RepertoireView` sortowanie (po ocenie / alfabetycznie) – użyj `SegmentedControl` i `useMemo`.
2. Dodaj `RefreshControl` do listy seansów (symulowane odświeżanie z `setTimeout`).
3. W `SeatMap` oznacz miejsca dla osób z niepełnosprawnością innym kolorem i etykietą dostępności.