Aplikacje mobilne

Zadanie do wykładu 3

Kino Pod Gwiazdami – pełna nawigacja aplikacji

Stack, Tabs, Drawer, Top Tabs, modale, deep linki, ochrona tras – React Navigation 7

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

Zamieniamy przełączanie widoków stanem na prawdziwą nawigację. Celem jest pokazanie **możliwie wszystkich** mechanizmów React Navigation 7 na jednej, realnej aplikacji: cztery rodzaje nawigatorów zagnieżdżone w sobie, różne sposoby prezentacji ekranów (karta, modal, przezroczysty modal, arkusz z wysokościami), opcje nagłówków, deep linki, ochronę tras dla niezalogowanych, blokadę wyjścia z niezapisanym wyborem i nawigację spoza komponentów.

Struktura: **Root Stack** (logowanie / aplikacja + modale) → **Drawer** (menu boczne) → **Bottom Tabs** (Repertuar, Seanse, Bilety, Profil) → w zakładkach **Stack** filmów oraz **Material Top Tabs** z dniami seansów.

> **Informacja**
> Gotowy kod tego etapu: katalog „kod/w03” obok instrukcji.

Co zbudujemy

- mapa nawigacji opisana typami TypeScript (każdy `navigate` i każdy parametr sprawdzany przy kompilacji),
- ochrona tras: niezalogowany użytkownik widzi tylko ekrany powitania i logowania (warunkowe renderowanie grup ekranów),
- Stack z różnymi prezentacjami: `card`, `modal`, `transparentModal` (zwiastun), `formSheet` z `sheetAllowedDetents` (filtry),
- nagłówki: duży tytuł iOS, wyszukiwarka w nagłówku, przezroczysty nagłówek z rozmyciem, przyciski w nagłówku, tytuł zależny od parametrów,
- dolne zakładki z ikonami, plakietką (badge), historią „Wstecz”, blokadą zakładki dla gościa (`tabPress` + `preventDefault`) i powrotem do listy po opuszczeniu zakładki (`popToTopOnBlur`),
- menu boczne (Drawer) z własną zawartością (`drawerContent`), przyciskiem wylogowania i otwieraniem z kodu (`DrawerActions`),
- górne zakładki z dniami seansów (przesuwane gestem, leniwe ładowanie),
- przekazywanie wyniku wstecz (filtry → lista) przez `popTo` z parametrami,
- blokada wyjścia z wyboru miejsc (`usePreventRemove`) z oknem potwierdzenia,
- deep linki (`kino://movie/m1`, `https://kino.example.com/movie/m1`) z konfiguracją ścieżek, parsowaniem parametrów i ekranem „Nie znaleziono”,
- `navigation.preload` – ekran filmu renderuje się w tle już przy dotknięciu karty,
- nawigacja spoza komponentów (`navigationRef`) i śledzenie ekranów (`onStateChange`).

Co z dokumentacji pokazujemy w praktyce

Temat z wykładu / dokumentacji	Gdzie w kodzie
NavigationContainer: `linking`, `theme`, `onReady`, `onStateChange`, `fallback`	`RootNavigator.tsx`, `navTheme.ts`, `linking.ts`
Native Stack: `Stack.Group`, `presentation` (modal, transparentModal, formSheet)	`RootNavigator.tsx`, `RepertoireStack.tsx`
formSheet: `sheetAllowedDetents`, `sheetGrabberVisible`, `sheetCornerRadius`	`RepertoireStack.tsx`

Temat z wykładu / dokumentacji	Gdzie w kodzie
Opcje nagłówka: `headerLargeTitle`, `headerSearchBarOptions`, `headerTransparent`, `headerBlurEffect`, `headerRight`, `headerBackButtonDisplayMode`	`RepertoireStack.tsx`, `MoviesScreen.tsx`, `MovieHeaderActions.tsx`
`options` jako funkcja (tytuł z parametru), `getId`, `animation`, `gestureEnabled`	`RootNavigator.tsx`, `RepertoireStack.tsx`
Bottom Tabs: `tabBarIcon`, `tabBarBadge`, `backBehavior`, `popToTopOnBlur`, `listeners`/`screenListeners`, `animation: "shift"`	`HomeTabs.tsx`
Drawer: `drawerContent`, `DrawerItemList`, `DrawerItem`, `drawerType`, `swipeEdgeWidth`	`MainDrawer.tsx`
Material Top Tabs: `initialParams`, `lazy`, `swipeEnabled`, `tabBarIndicatorStyle`	`ScreeningsTopTabs.tsx`
Akcje: `navigate` (zagnieżdżone), `push`, `replace`, `popTo`, `goBack`, `reset`, `setOptions`, `preload`	ekrany w `src/screens/…`
Hooki: `usePreventRemove`, `useScrollToTop`, `useIsFocused`, `useNavigation`	`SeatSelectionScreen.tsx`, `MoviesScreen.tsx`, `ScreeningsDayScreen.tsx`
Typy: `NativeStackScreenProps`, `CompositeScreenProps`, `NavigatorScreenParams`, globalny `RootParamList`	`navigation/types.ts`
Deep linking: `prefixes`, zagnieżdżony `config`, `parse`/`stringify`, `*` (NotFound), `Linking.createURL`	`linking.ts`, `AboutScreen.tsx`
Ochrona tras przez warunkowe renderowanie ekranów	`RootNavigator.tsx`, `AuthContext.tsx`
`createNavigationContainerRef` – nawigacja spoza drzewa komponentów	`navigationRef.ts`

Przed rozpoczęciem

Ukończony etap W02 (projekt z katalogu `kod/w02`).

Instalacja pakietów

Instalujemy React Navigation 7 z czterema rodzajami nawigatorów oraz biblioteki natywne, których wymagają (ekrany natywne, gesty, pager dla górnych zakładek):

Terminal

```
npx expo install @react-navigation/native @react-navigation/native-stack @react-navigation/bottom-tabs @react-navigation/drawer @react-navigation/material-top-tabs @react-navigation/elements react-native-screens react-native-safe-area-context react-native-gesture-handler react-native-reanimated react-native-worklets react-native-pager-view expo-linking
```

> **Informacja**
>
> Drawer wymaga `react-native-gesture-handler` i `react-native-reanimated` (w SDK 58 Reanimated 4 + `react-native-worklets`), a górne zakładki – `react-native-pager-view`. Wszystkie są dostępne w Expo Go.

> **Uwaga**
> W `app.json` dodajemy `"scheme": "kino"` – bez schematu deep linki `kino://…` nie zadziałają w buildzie (w Expo Go adresy mają postać `exp://IP:8081/--/…`).

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
`src/auth/AuthContext.tsx`	NOWY	minimalny stan logowania (rozbudowa: W04, W06, W10)
`src/components/MovieHeaderActions.tsx`	NOWY	przyciski w nagłówku ekranu szczegółów filmu
`src/navigation/HomeTabs.tsx`	NOWY	dolne zakładki
`src/navigation/linking.ts`	NOWY	deep linki: adres URL -> ekran (z parametrami)
`src/navigation/MainDrawer.tsx`	NOWY	menu boczne z własną zawartością
`src/navigation/navigationRef.ts`	NOWY	nawigacja spoza komponentów (np. z powiadomień w W07)
`src/navigation/navTheme.ts`	NOWY	motyw React Navigation zbudowany z motywu aplikacji
`src/navigation/RepertoireStack.tsx`	NOWY	Stack zagnieżdżony w zakładce „Repertuar”
`src/navigation/RootNavigator.tsx`	NOWY	kontener nawigacji + główny Stack z ochroną tras
`src/navigation/ScreeningsTopTabs.tsx`	NOWY	górne zakładki (przesuwane gestem) z dniami
`src/navigation/types.ts`	NOWY	mapa całej nawigacji aplikacji w typach
`src/screens/auth/SignInScreen.tsx`	NOWY	uproszczone logowanie (pełny formularz w W05)
`src/screens/auth/WelcomeScreen.tsx`	NOWY	ekran powitalny (tylko dla niezalogowanych)
`src/screens/booking/SeatSelectionScreen.tsx`	NOWY	wybór miejsc (modal) z blokadą wyjścia
`src/screens/drawer/AboutScreen.tsx`	NOWY	informacje + linki wychodzące (Linking)
`src/screens/drawer/HallsScreen.tsx`	NOWY	lista sal (ekran menu bocznego)
`src/screens/drawer/SettingsScreen.tsx`	NOWY	ustawienia (działające opcje dodamy w W04)
`src/screens/movies/FiltersScreen.tsx`	NOWY	arkusz (formSheet) przekazujący wynik wstecz
`src/screens/movies/MovieDetailsScreen.tsx`	NOWY	szczegóły filmu + lista seansów
`src/screens/movies/MoviesScreen.tsx`	NOWY	repertuar: wyszukiwarka w nagłówku, filtry w arkuszu
`src/screens/movies/TrailerScreen.tsx`	NOWY	przezroczysty modal nad poprzednim ekranem
`src/screens/NotFoundScreen.tsx`	NOWY	nieznany deep link
`src/screens/profile/ProfileScreen.tsx`	NOWY	profil + przykłady nawigacji między nawigatorami

Plik	Stan	Opis
src/screens/screenings/ScreeningsDayScreen.tsx	NOWY	seanse jednego dnia (górna zakładka)
src/screens/tickets/TicketDetailsScreen.tsx	NOWY	szczegóły biletu + nawigacja zagnieżdżona
src/screens/tickets/TicketsScreen.tsx	NOWY	lista kupionych biletów
src/state/TicketsContext.tsx	NOWY	bilety kupione w trakcie sesji (w W04 zastąpi je store)
app.json	ZMIENIONY	
App.tsx	ZMIENIONY	etap W03: providery + nawigacja
src/components/MovieCard.tsx	ZMIENIONY	kafelek filmu w siatce
src/components/ScreeningRow.tsx	ZMIENIONY	wiersz seansu w SectionList
src/components/SegmentedControl.tsx	ZMIENIONY	przełącznik widoków (do W03 zastępuje nawigację)
src/data/mockData.ts	ZMIENIONY	dane przykładowe (od W06 zastąpi je API)
src/data/types.ts	ZMIENIONY	modele domenowe aplikacji
src/utils/format.ts	ZMIENIONY	formatowanie przez wbudowane Intl (bez dodatkowych bibliotek)
src/components/MovieDetailsModal.tsx	USUŃ	
src/screens/HallView.tsx	USUŃ	
src/screens/RepertoireView.tsx	USUŃ	
src/screens/ScreeningsView.tsx	USUŃ	

Krok po kroku

1. Konfiguracja: schemat deep linków

Do `app.json` dochodzi `scheme` (deep linki) – reszta bez zmian.

app.json

Zastąp całą zawartość pliku

```
{
  "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"
    ],
    "scheme": "kino"
  }
}
```

2. Typy całej nawigacji

Zaczynamy od typów – to „mapa" aplikacji. Każdy nawigator ma swoją listę ekranów i parametrów (`…ParamList`). Zagnieżdżony nawigator jest parametrem rodzica (`NavigatorScreenParams`). Typy propsów ekranów składamy od dołu przez `CompositeScreenProps`, dzięki czemu z ekranu w zakładce można nawigować do ekranu w Root Stacku.

Deklaracja `ReactNavigation.RootParamList` sprawia, że także `useNavigation()` bez generyka zna wszystkie ekrany.

src/navigation/types.ts

Nowy plik – mapa całej nawigacji aplikacji w typach

```
// src/navigation/types.ts – mapa całej nawigacji aplikacji w typach
import type { CompositeScreenProps, NavigatorScreenParams } from "@react-navigation/native";
import type { NativeStackScreenProps } from "@react-navigation/native-stack";
import type { BottomTabScreenProps } from "@react-navigation/bottom-tabs";
import type { DrawerScreenProps } from "@react-navigation/drawer";
import type { MaterialTopTabScreenProps } from "@react-navigation/material-top-tabs";
import type { Genre } from "@/data/types";

// ── Poziom 1: główny Stack ──────────────────────────────────
export type RootStackParamList = {
  Welcome: undefined;
  SignIn: { email?: string } | undefined;
  Main: NavigatorScreenParams<DrawerParamList> | undefined;
  SeatSelection: { screeningId: string };
  TicketDetails: { ticketId: string };
  Trailer: { movieId: string };
```

```
  NotFound: undefined;
};

// ── Poziom 2: Drawer (menu boczne) ──────────────────────────────
export type DrawerParamList = {
  HomeTabs: NavigatorScreenParams<TabParamList> | undefined;
  Halls: undefined;
  Settings: undefined;
  About: undefined;
};

// ── Poziom 3: dolne zakładki ────────────────────────────────────
export type TabParamList = {
  RepertoireTab: NavigatorScreenParams<RepertoireStackParamList> | undefined;
  ScreeningsTab: NavigatorScreenParams<ScreeningsTopTabParamList> | undefined;
  TicketsTab: undefined;
  ProfileTab: undefined;
};

// ── Poziom 4a: Stack wewnątrz zakładki „Repertuar” ──────────────
export type MovieFilters = { genres?: Genre[]; onlyPremieres?: boolean };

export type RepertoireStackParamList = {
  Movies: MovieFilters | undefined;
  MovieDetails: { movieId: string };
  Filters: MovieFilters | undefined;
};

// ── Poziom 4b: górne zakładki w „Seansach” ──────────────────────
export type ScreeningsTopTabParamList = {
  Today: { dayOffset: number };
  Tomorrow: { dayOffset: number };
  DayAfter: { dayOffset: number };
};

// ── Typy propsów ekranów (łączą nawigatory od dołu do góry) ─────
export type RootStackScreenProps<T extends keyof RootStackParamList> = NativeStackScreenProps<
  RootStackParamList,
  T
>;

export type DrawerScreenPropsOf<T extends keyof DrawerParamList> = CompositeScreenProps<
  DrawerScreenProps<DrawerParamList, T>,
  RootStackScreenProps<keyof RootStackParamList>
>;

export type TabScreenProps<T extends keyof TabParamList> = CompositeScreenProps<
  BottomTabScreenProps<TabParamList, T>,
  DrawerScreenPropsOf<keyof DrawerParamList>
>;

export type RepertoireScreenProps<T extends keyof RepertoireStackParamList> =
CompositeScreenProps<
  NativeStackScreenProps<RepertoireStackParamList, T>,
  TabScreenProps<keyof TabParamList>
>;

export type ScreeningsTopTabScreenProps<T extends keyof ScreeningsTopTabParamList> =
  CompositeScreenProps<
    MaterialTopTabScreenProps<ScreeningsTopTabParamList, T>,
    TabScreenProps<keyof TabParamList>
  >;

// Domyślny typ dla useNavigation() bez generyka
```

```
declare global {
  namespace ReactNavigation {
    // eslint-disable-next-line @typescript-eslint/no-empty-object-type
    interface RootParamList extends RootStackParamList {}
  }
}
```

3. Zmiany w modelu danych i komponentach

Dane przykładowe i typy dostają drobne rozszerzenia potrzebne ekranom (np. identyfikatory w adresach), a karta filmu i wiersz seansu – obsługę `onPressIn` (wcześniejsze przygotowanie ekranu).

src/data/types.ts

Zastąp całą zawartość pliku – 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

Zastąp całą zawartość pliku – 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);
```

src/utils/format.ts

Zastąp całą zawartość pliku – 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}`;
```

src/components/MovieCard.tsx

Zastąp całą zawartość pliku – 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;
  onPressIn?: (movie: Movie) => void;
};

function MovieCardComponent({ movie, onPress, onLongPress, onPressIn }: Props) {
  const theme = useAppTheme();
  return (
    <Pressable
      onPress={() => onPress(movie)}
      onPressIn={() => onPressIn?.(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/ScreeningRow.tsx

Zastąp całą zawartość pliku – 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/components/SegmentedControl.tsx

Zastąp całą zawartość pliku – 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,
  },
});
```

4. Stan logowania i bilety sesji (Context)

Na tym etapie potrzebujemy tylko informacji „zalogowany / gość / niezalogowany” – prosty `AuthContext`. Bilety kupione w trakcie sesji trzyma `TicketsContext` (na wykładzie 4 zastąpimy go magazynem stanu, a na 6 – API).

src/auth/AuthContext.tsx

Nowy plik – minimalny stan logowania (rozbudowa: W04, W06, W10)

```
// src/auth/AuthContext.tsx – minimalny stan logowania (rozbudowa: W04, W06, W10)
import { createContext, useContext, useMemo, useState, type ReactNode } from "react";

export type User = { name: string; email?: string; isGuest: boolean };

type AuthContextValue = {
  user: User | null;
  signIn: (user: User) => void;
  signOut: () => void;
};

const AuthContext = createContext<AuthContextValue | null>(null);

export function AuthProvider({ children }: { children: ReactNode }) {
  const [user, setUser] = useState<User | null>(null);

  const value = useMemo(
    () => ({ user, signIn: (u: User) => setUser(u), signOut: () => setUser(null) }),
    [user],
  );

  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
}

export function useAuth() {
  const ctx = useContext(AuthContext);
  if (!ctx) throw new Error("useAuth musi być użyte wewnątrz <AuthProvider>");
  return ctx;
}
```

src/state/TicketsContext.tsx

Nowy plik – bilety kupione w trakcie sesji (w W04 zastąpi je store)

```
// src/state/TicketsContext.tsx – bilety kupione w trakcie sesji (w W04 zastąpi je store)
import { createContext, useContext, useMemo, useState, type ReactNode } from "react";

export type Ticket = {
  id: string;
  screeningId: string;
  seats: string[];
  total: number;
  createdAt: string;
};

type TicketsContextValue = {
  tickets: Ticket[];
  addTicket: (ticket: Omit<Ticket, "id" | "createdAt">) => Ticket;
};

const TicketsContext = createContext<TicketsContextValue | null>(null);

export function TicketsProvider({ children }: { children: ReactNode }) {
```

```
  const [tickets, setTickets] = useState<Ticket[]>([]);

  const value = useMemo<TicketsContextValue>(
    () => ({
      tickets,
      addTicket: (data) => {
        const ticket: Ticket = {
          ...data,
          id: `T${Date.now().toString(36).toUpperCase()}`,
          createdAt: new Date().toISOString(),
        };
        setTickets((prev) => [ticket, ...prev]);
        return ticket;
      },
    }),
    [tickets],
  );

  return <TicketsContext.Provider value={value}>{children}</TicketsContext.Provider>;
}

export function useTickets() {
  const ctx = useContext(TicketsContext);
  if (!ctx) throw new Error("useTickets musi być użyte wewnątrz <TicketsProvider>");
  return ctx;
}
```

5. Motyw nawigacji i referencja do nawigatora

`navTheme.ts` buduje motyw React Navigation z kolorów aplikacji (nagłówki, tła, zakładki w trybie jasnym i ciemnym). `navigationRef` pozwala nawigować spoza komponentów – przyda się przy powiadomieniach (wykład 7).

src/navigation/navTheme.ts

Nowy plik – motyw React Navigation zbudowany z motywu aplikacji

```
// src/navigation/navTheme.ts – motyw React Navigation zbudowany z motywu aplikacji
import { DarkTheme, DefaultTheme, type Theme } from "@react-navigation/native";
import { fonts, type AppTheme } from "@/theme/theme";

export function toNavigationTheme(theme: AppTheme): Theme {
  const base = theme.dark ? DarkTheme : DefaultTheme;
  return {
    ...base,
    colors: {
      ...base.colors,
      primary: theme.colors.primary, // aktywne elementy (np. zakładka)
      background: theme.colors.background, // tło ekranów
      card: theme.colors.surface, // nagłówki, pasek zakładek, drawer
      text: theme.colors.text,
      border: theme.colors.border,
      notification: theme.colors.primary, // kolor badge
    },
    // W v7 motyw zawiera też fonty używane w nagłówkach i zakładkach
    fonts: {
      regular: { fontFamily: fonts.regular, fontWeight: "400" },
      medium: { fontFamily: fonts.medium, fontWeight: "500" },
      bold: { fontFamily: fonts.bold, fontWeight: "700" },
      heavy: { fontFamily: fonts.black, fontWeight: "900" },
    },
  };
}
```

src/navigation/navigationRef.ts

Nowy plik – nawigacja spoza komponentów (np. z powiadomień w W07)

```
// src/navigation/navigationRef.ts – nawigacja spoza komponentów (np. z powiadomień w W07)
import { createNavigationContainerRef } from "@react-navigation/native";
import type { RootStackParamList } from "./types";

export const navigationRef = createNavigationContainerRef<RootStackParamList>();

// Bezpieczne wywołanie: nawigator może jeszcze nie być gotowy
export function navigateToTicket(ticketId: string) {
  if (navigationRef.isReady()) {
    navigationRef.navigate("TicketDetails", { ticketId });
  }
}
```

6. Stack filmów w zakładce „Repertuar”

Lista filmów ma duży tytuł iOS i wyszukiwarkę w nagłówku. Szczegóły filmu mają przezroczysty nagłówek z rozmyciem i przyciski (udostępnij, ulubione). `getId` sprawia, że inny film to nowy ekran (a nie podmiana parametrów). Filtry otwierają się jako **arkusz** (`formSheet`) z dwiema wysokościami i uchwytem.

src/navigation/RepertoireStack.tsx

Nowy plik – Stack zagnieżdżony w zakładce „Repertuar”

```
// src/navigation/RepertoireStack.tsx – Stack zagnieżdżony w zakładce „Repertuar”
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { DrawerToggleButton } from "@react-navigation/drawer";
import { useAppTheme } from "@/theme/theme";
import type { RepertoireStackParamList } from "./types";
import { MoviesScreen } from "@/screens/movies/MoviesScreen";
import { MovieDetailsScreen } from "@/screens/movies/MovieDetailsScreen";
import { FiltersScreen } from "@/screens/movies/FiltersScreen";
import { MovieHeaderActions } from "@/components/MovieHeaderActions";

const Stack = createNativeStackNavigator<RepertoireStackParamList>();

export function RepertoireStack() {
  const theme = useAppTheme();
  return (
    <Stack.Navigator
      screenOptions={{
        headerTintColor: theme.colors.primary,
        headerBackButtonDisplayMode: "minimal",
        headerTitleAlign: "center", // Android: tytuł na środku (iOS – zawsze)
      }}
    >
      <Stack.Screen
        name="Movies"
        component={MoviesScreen}
        options={{
          title: "Repertuar",
          headerLargeTitle: true, // iOS: duży tytuł zwijany przy przewijaniu
          headerLeft: () => <DrawerToggleButton tintColor={theme.colors.primary} />,
        }}
      />
      <Stack.Screen
        name="MovieDetails"
        component={MovieDetailsScreen}
        // Unikalny identyfikator: kolejny film to nowy ekran, nie aktualizacja parametrów
        getId={({ params }) => params.movieId}
```

```
        // options jako funkcja – dostęp do route (parametrów ekranu)
        options={({ route }) => ({
          title: "",
          headerTransparent: true, // treść wchodzi pod nagłówek
          headerBlurEffect: "systemChromeMaterial", // iOS: rozmyte tło nagłówka
          headerTintColor: "#FFFFFF",
          headerRight: () => <MovieHeaderActions movieId={route.params.movieId} />,
        })}
      />
      <Stack.Screen
        name="Filters"
        component={FiltersScreen}
        options={{
          title: "Filtry",
          presentation: "formSheet", // arkusz od dołu
          sheetAllowedDetents: [0.5, 1], // pół ekranu lub cały ekran
          sheetGrabberVisible: true, // uchwyt do przeciągania
          sheetCornerRadius: 24,
        }}
      />
    </Stack.Navigator>
  );
}
```

src/components/MovieHeaderActions.tsx

Nowy plik – przyciski w nagłówku ekranu szczegółów filmu

```
// src/components/MovieHeaderActions.tsx – przyciski w nagłówku ekranu szczegółów filmu
import { useState } from "react";
import { Share, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { HeaderButton } from "@react-navigation/elements";
import { findMovie } from "@/data/mockData";

// Komponent przekazujemy w `options` nawigatora (a nie przez setOptions z ekranu),
// dzięki czemu działa też z navigation.preload() – ekran wczytany zawczasu
// nie może wywoływać setOptions.
export function MovieHeaderActions({ movieId }: { movieId: string }) {
  const movie = findMovie(movieId);
  const [favorite, setFavorite] = useState(false); // w W04 przeniesiemy do globalnego store

  return (
    <View style={styles.row}>
      <HeaderButton
        accessibilityLabel="Udostępnij"
        onPress={() =>
          movie && Share.share({ message: `Idę na „${movie.title}”! kino://movie/${movie.id}` })
        }
      >
        <Ionicons name="share-outline" size={22} color="#FFFFFF" />
      </HeaderButton>
      <HeaderButton
        accessibilityLabel={favorite ? "Usuń z ulubionych" : "Dodaj do ulubionych"}
        onPress={() => setFavorite((f) => !f)}
      >
        <Ionicons name={favorite ? "heart" : "heart-outline"} size={22} color="#FFFFFF" />
      </HeaderButton>
    </View>
  );
}

const styles = StyleSheet.create({ row: { flexDirection: "row" } });
```

7. Ekrany repertuaru

`MoviesScreen` ustawia wyszukiwarkę w nagłówku przez `navigation.setOptions` w `useLayoutEffect` i przewija listę na górę po ponownym dotknięciu zakładki (`useScrollToTop`). `FiltersScreen` zwraca wynik do listy przez `navigation.popTo("Movies", { …filtry })`. `MovieDetailsScreen` pokazuje seanse i otwiera wybór miejsc w Root Stacku. `TrailerScreen` to przezroczysty modal – widać pod nim poprzedni ekran.

src/screens/movies/MoviesScreen.tsx

Nowy plik – repertuar: wyszukiwarka w nagłówku, filtry w arkuszu

```
// src/screens/movies/MoviesScreen.tsx – repertuar: wyszukiwarka w nagłówku, filtry w arkuszu
import { useCallback, useLayoutEffect, useMemo, useRef, useState } from "react";
import { FlatList, Pressable, StyleSheet, View, useWindowDimensions } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { useScrollToTop } from "@react-navigation/native";
import { MOVIES } from "@/data/mockData";
import type { Movie } from "@/data/types";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { MovieCard } from "@/components/MovieCard";
import { PremiereCarousel } from "@/components/PremiereCarousel";
import type { RepertoireScreenProps } from "@/navigation/types";
import { spacing, useAppTheme } from "@/theme/theme";

export function MoviesScreen({ navigation, route }: RepertoireScreenProps<"Movies">) {
  const theme = useAppTheme();
  const { width } = useWindowDimensions();
  const [query, setQuery] = useState("");
  const listRef = useRef<FlatList<Movie>>(null);

  // Ponowne dotknięcie aktywnej zakładki przewija listę na górę
  useScrollToTop(listRef);

  // Filtry przechowujemy w parametrach trasy – ustawia je ekran Filters
  // useMemo: bez niego „?? []” tworzyłoby nową tablicę przy każdym renderze
  const genres = useMemo(() => route.params?.genres ?? [], [route.params?.genres]);
  const onlyPremieres = route.params?.onlyPremieres ?? false;
  const activeFilters = genres.length + (onlyPremieres ? 1 : 0);

  // Opcje nagłówka zależne od stanu ekranu ustawiamy przez setOptions
  useLayoutEffect(() => {
    navigation.setOptions({
      headerSearchBarOptions: {
        placeholder: "Szukaj filmu",
        onChangeText: (e) => setQuery(e.nativeEvent.text),
        hideWhenScrolling: true,
      },
      headerRight: () => (
        <Pressable
          onPress={() => navigation.navigate("Filters", { genres, onlyPremieres })}
          hitSlop={10}
          accessibilityRole="button"
          accessibilityLabel={`Filtry, aktywne: ${activeFilters}`}
          style={styles.filterButton}
        >
          <Ionicons name="options-outline" size={22} color={theme.colors.primary} />
          {activeFilters > 0 && (
            <View style={[styles.badge, { backgroundColor: theme.colors.primary }]}>
              <AppText variant="caption" color={theme.colors.onPrimary}>
                {activeFilters}
              </AppText>
            </View>
          )}
```

```
        </Pressable>
      ),
    });
  }, [navigation, genres, onlyPremieres, activeFilters, theme]);

  const numColumns = width >= 900 ? 4 : width >= 600 ? 3 : 2;
  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 openMovie = useCallback(
    (movie: Movie) => navigation.navigate("MovieDetails", { movieId: movie.id }),
    [navigation],
  );
  // preload (v7): ekran szczegółów renderuje się w tle już przy dotknięciu karty
  const preloadMovie = useCallback(
    (movie: Movie) => navigation.preload("MovieDetails", { movieId: movie.id }),
    [navigation],
  );

  return (
    <FlatList
      ref={listRef}
      key={numColumns}
      data={filtered}
      keyExtractor={(item) => item.id}
      numColumns={numColumns}
      // iOS: lista współpracuje z dużym tytułem i paskiem wyszukiwania
      contentInsetAdjustmentBehavior="automatic"
      renderItem={({ item }) => (
        <View style={[styles.cell, { maxWidth: `${100 / numColumns}%` }]}>
          <MovieCard movie={item} onPress={openMovie} onPressIn={preloadMovie} />
        </View>
      )}
      ListHeaderComponent={
        query === "" ? (
          <View style={styles.header}>
            <PremiereCarousel movies={MOVIES.filter((m) => m.isPremiere)} onPress={openMovie} />
            <AppText variant="label" muted style={styles.count}>
              Filmy: {filtered.length}
            </AppText>
          </View>
        ) : undefined
      }
      ListEmptyComponent={<EmptyState title="Brak filmów" message="Zmień frazę lub filtry." />}
      columnWrapperStyle={styles.columns}
      contentContainerStyle={styles.content}
      keyboardDismissMode="on-drag"
    />
  );
}

const styles = StyleSheet.create({
  header: { gap: spacing.md, paddingTop: spacing.md, marginBottom: spacing.md },
  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 },
  filterButton: { padding: 4 },
  badge: {
    position: "absolute",
    top: -4,
```

```
    right: -6,
    minWidth: 18,
    height: 18,
    borderRadius: 9,
    alignItems: "center",
    justifyContent: "center",
    paddingHorizontal: 4,
  },
});
```

src/screens/movies/FiltersScreen.tsx

Nowy plik – arkusz (formSheet) przekazujący wynik wstecz

```
// src/screens/movies/FiltersScreen.tsx – arkusz (formSheet) przekazujący wynik wstecz
import { useState } from "react";
import { Pressable, StyleSheet, Switch, View } from "react-native";
import { GENRES } from "@/data/mockData";
import type { Genre } from "@/data/types";
import { AppText } from "@/components/AppText";
import { GenreChips } from "@/components/GenreChips";
import type { RepertoireScreenProps } from "@/navigation/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function FiltersScreen({ navigation, route }: RepertoireScreenProps<"Filters">) {
  const theme = useAppTheme();
  const [genres, setGenres] = useState<Genre[]>(route.params?.genres ?? []);
  const [onlyPremieres, setOnlyPremieres] = useState(route.params?.onlyPremieres ?? false);

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

  // popTo + merge: wracamy do ekranu Movies i scalamy nowe parametry z obecnymi
  const apply = () => navigation.popTo("Movies", { genres, onlyPremieres }, { merge: true });

  return (
    <View style={styles.container}>
      <AppText variant="label" muted>
        Gatunki
      </AppText>
      <View style={styles.chips}>
        <GenreChips genres={GENRES} selected={genres} onToggle={toggle} />
      </View>
      <View style={styles.row}>
        <AppText>Tylko premiery</AppText>
        <Switch value={onlyPremieres} onValueChange={setOnlyPremieres} />
      </View>
      <View style={styles.actions}>
        <Pressable
          style={[styles.button, { borderColor: theme.colors.border }]}
          onPress={() => {
            setGenres([]);
            setOnlyPremieres(false);
          }}
        >
          <AppText>Wyczyść</AppText>
        </Pressable>
        <Pressable
          style={[styles.button, { backgroundColor: theme.colors.primary }]}
          onPress={apply}
        >
          <AppText color={theme.colors.onPrimary}>Zastosuj</AppText>
        </Pressable>
      </View>
    </View>
  );
```

```
}

const styles = StyleSheet.create({
  container: { padding: spacing.lg, gap: spacing.lg },
  chips: { marginHorizontal: -spacing.lg },
  row: { flexDirection: "row", justifyContent: "space-between", alignItems: "center" },
  actions: { flexDirection: "row", gap: spacing.md },
  button: {
    flex: 1,
    height: 48,
    borderRadius: radius.md,
    borderWidth: 1,
    borderColor: "transparent",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

src/screens/movies/MovieDetailsScreen.tsx

Nowy plik – szczegóły filmu + lista seansów

```
// src/screens/movies/MovieDetailsScreen.tsx – szczegóły filmu + lista seansów
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { LinearGradient } from "expo-linear-gradient";
import { Ionicons } from "@expo/vector-icons";
import { SCREENINGS, findHall, findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { AgeBadge } from "@/components/AgeBadge";
import { EmptyState } from "@/components/EmptyState";
import { RatingStars } from "@/components/RatingStars";
import { ScreeningRow } from "@/components/ScreeningRow";
import type { RepertoireScreenProps } from "@/navigation/types";
import { formatDuration } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function MovieDetailsScreen({ navigation, route }: RepertoireScreenProps<"MovieDetails">)
{
  const theme = useAppTheme();
  const movie = findMovie(route.params.movieId);

  if (!movie) {
    return (
      <EmptyState title="Nie znaleziono filmu" message="Film mógł zostać zdjęty z repertuaru."
/>
    );
  }

  const screenings = SCREENINGS.filter((s) => s.movieId === movie.id);

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View>
        <Image source={movie.backdropUrl} style={styles.backdrop} contentFit="cover" />
        <LinearGradient
          colors={["rgba(0,0,0,0.6)", "transparent", theme.colors.background]}
          style={StyleSheet.absoluteFill}
        />
      </View>
      <View style={styles.body}>
        <AppText variant="hero">{movie.title}</AppText>
        <View style={styles.meta}>
          <AgeBadge rating={movie.ageRating} />
          <AppText muted>{formatDuration(movie.durationMin)}</AppText>
          <RatingStars rating={movie.rating} />
```

```
        </View>
        <AppText>{movie.description}</AppText>

        <Pressable
          // Ekran Trailer jest w głównym Stacku – navigate „szuka” go w nawigatorach
nadrzędnych
          onPress={() => navigation.navigate("Trailer", { movieId: movie.id })}
          style={[styles.trailer, { borderColor: theme.colors.primary }]}
        >
          <Ionicons name="play-circle" size={22} color={theme.colors.primary} />
          <AppText variant="subtitle" color={theme.colors.primary}>
            Zwiastun
          </AppText>
        </Pressable>

        <AppText variant="title">Seanse</AppText>
      </View>
      {screenings.length === 0 ? (
        <EmptyState icon="calendar-outline" title="Brak seansów" />
      ) : (
        screenings.map((s) => {
          const hall = findHall(s.hallId);
          return hall ? (
            <View key={s.id} style={styles.screening}>
              <ScreeningRow
                screening={s}
                movie={movie}
                hall={hall}
                onPress={() => navigation.navigate("SeatSelection", { screeningId: s.id })}
              />
            </View>
          ) : null;
        })
      )}
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  content: { paddingBottom: spacing.xxl },
  backdrop: { width: "100%", aspectRatio: 4 / 3 },
  body: { padding: spacing.lg, gap: spacing.md },
  meta: { flexDirection: "row", alignItems: "center", gap: spacing.md, flexWrap: "wrap" },
  trailer: {
    flexDirection: "row",
    alignItems: "center",
    alignSelf: "flex-start",
    gap: spacing.sm,
    paddingHorizontal: spacing.lg,
    paddingVertical: spacing.sm,
    borderWidth: 1.5,
    borderRadius: radius.pill,
  },
  screening: { marginBottom: spacing.sm },
});
```

src/screens/movies/TrailerScreen.tsx

Nowy plik – przezroczysty modal nad poprzednim ekranem

```
// src/screens/movies/TrailerScreen.tsx – przezroczysty modal nad poprzednim ekranem
import { Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { Ionicons } from "@expo/vector-icons";
import { findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
```

```
import type { RootStackScreenProps } from "@/navigation/types";
import { radius, spacing } from "@/theme/theme";

export function TrailerScreen({ navigation, route }: RootStackScreenProps<"Trailer">) {
  const movie = findMovie(route.params.movieId);

  return (
    <View style={styles.overlay}>
      {/* Dotknięcie tła zamyka modal */}
      <Pressable style={StyleSheet.absoluteFill} onPress={() => navigation.goBack()} />
      <View style={styles.card}>
        <Image source={movie?.backdropUrl} style={styles.video} contentFit="cover" />
        <View style={styles.play}>
          <Ionicons name="play" size={36} color="#FFFFFF" />
        </View>
        <AppText variant="subtitle" color="#FFFFFF" style={styles.title}>
          {movie?.title ?? "Zwiastun"}
        </AppText>
        <AppText variant="caption" color="#D1D1D6" style={styles.title}>
          Odtwarzacz wideo dodamy w W07 – teraz pokazujemy nawigację modalną.
        </AppText>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  overlay: {
    flex: 1,
    backgroundColor: "rgba(0,0,0,0.7)",
    justifyContent: "center",
    padding: spacing.lg,
  },
  card: { backgroundColor: "#111114", borderRadius: radius.lg, overflow: "hidden" },
  video: { width: "100%", aspectRatio: 16 / 9 },
  play: {
    position: "absolute",
    top: 0,
    left: 0,
    right: 0,
    aspectRatio: 16 / 9,
    alignItems: "center",
    justifyContent: "center",
  },
  title: { paddingHorizontal: spacing.lg, paddingTop: spacing.md },
});
```

8. Górne zakładki z dniami seansów

Material Top Tabs: trzy zakładki (Dziś, Jutro, Pojutrze) korzystają z tego samego komponentu ekranu z różnym `initialParams`. Zakładki przesuwa się gestem; ekrany ładują się leniwie (`lazy`).

src/navigation/ScreeningsTopTabs.tsx

Nowy plik – górne zakładki (przesuwane gestem) z dniami

```
// src/navigation/ScreeningsTopTabs.tsx – górne zakładki (przesuwane gestem) z dniami
import { ActivityIndicator } from "react-native";
import { createMaterialTopTabNavigator } from "@react-navigation/material-top-tabs";
import { fonts, useAppTheme } from "@/theme/theme";
import type { ScreeningsTopTabParamList } from "./types";
import { ScreeningsDayScreen } from "@/screens/screenings/ScreeningsDayScreen";

const TopTab = createMaterialTopTabNavigator<ScreeningsTopTabParamList>();
```

```
export function ScreeningsTopTabs() {
  const theme = useAppTheme();
  return (
    <TopTab.Navigator
      screenOptions={{
        tabBarActiveTintColor: theme.colors.primary,
        tabBarInactiveTintColor: theme.colors.textMuted,
        tabBarIndicatorStyle: { backgroundColor: theme.colors.primary, height: 3 },
        tabBarLabelStyle: { fontFamily: fonts.semibold, textTransform: "none" },
        swipeEnabled: true, // przełączanie przesunięciem palca
        lazy: true, // ekran tworzony dopiero przy pierwszym wejściu
        lazyPlaceholder: () => <ActivityIndicator style={{ marginTop: 32 }} />,
      }}
    >
      <TopTab.Screen
        name="Today"
        component={ScreeningsDayScreen}
        initialParams={{ dayOffset: 0 }} // parametr początkowy ekranu
        options={{ title: "Dziś" }}
      />
      <TopTab.Screen
        name="Tomorrow"
        component={ScreeningsDayScreen}
        initialParams={{ dayOffset: 1 }}
        options={{ title: "Jutro" }}
      />
      <TopTab.Screen
        name="DayAfter"
        component={ScreeningsDayScreen}
        initialParams={{ dayOffset: 2 }}
        options={{ title: "Pojutrze" }}
      />
    </TopTab.Navigator>
  );
}
```

src/screens/screenings/ScreeningsDayScreen.tsx

Nowy plik – seanse jednego dnia (górna zakładka)

```
// src/screens/screenings/ScreeningsDayScreen.tsx – seanse jednego dnia (górna zakładka)
import { useMemo } from "react";
import { FlatList, StyleSheet, View } from "react-native";
import { useIsFocused } from "@react-navigation/native";
import { SCREENINGS, findHall, findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { ScreeningRow } from "@/components/ScreeningRow";
import type { ScreeningsTopTabScreenProps, ScreeningsTopTabParamList } from
"@/navigation/types";
import { formatDayLabel, startOfDay } from "@/utils/format";
import { spacing } from "@/theme/theme";

type Props = ScreeningsTopTabScreenProps<keyof ScreeningsTopTabParamList>;

export function ScreeningsDayScreen({ navigation, route }: Props) {
  const { dayOffset } = route.params;
  const isFocused = useIsFocused(); // czy ta zakładka jest aktualnie widoczna

  const day = useMemo(() => {
    const d = startOfDay(new Date());
    d.setDate(d.getDate() + dayOffset);
    return d;
  }, [dayOffset]);

  const screenings = useMemo(
```

```
    () =>
      SCREENINGS.filter((s) => startOfDay(new Date(s.startsAt)).getTime() ===
day.getTime()).sort(
        (a, b) => a.startsAt.localeCompare(b.startsAt),
      ),
    [day],
  );

  return (
    <FlatList
      data={screenings}
      keyExtractor={(s) => s.id}
      renderItem={({ item }) => {
        const movie = findMovie(item.movieId);
        const hall = findHall(item.hallId);
        return movie && hall ? (
          <ScreeningRow
            screening={item}
            movie={movie}
            hall={hall}
            onPress={() => navigation.navigate("SeatSelection", { screeningId: item.id })}
          />
        ) : null;
      }}
      ListHeaderComponent={
        <View style={styles.header}>
          <AppText variant="title">{formatDayLabel(day.toISOString())}</AppText>
          {isFocused && (
            <AppText variant="caption" muted>
              Przesuń w bok, aby zmienić dzień
            </AppText>
          )}
        </View>
      }
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      ListEmptyComponent={<EmptyState icon="calendar-outline" title="Brak seansów tego dnia" />}
      contentContainerStyle={styles.content}
    />
  );
}

const styles = StyleSheet.create({
  header: { padding: spacing.lg, gap: 2 },
  separator: { height: spacing.sm },
  content: { paddingBottom: spacing.xxl },
});
```

9. Bilety i profil

`TicketDetailsScreen` pokazuje nawigację do zagnieżdżonego ekranu w innej zakładce (pełna ścieżka `navigate("Main", { screen: …, params: … })`) i `reset` historii. `ProfileScreen` zbiera przykłady przejść między nawigatorami: do zakładki, do ekranu w Stacku innej zakładki, otwarcie Drawera akcją `DrawerActions.openDrawer()`.

src/screens/tickets/TicketsScreen.tsx

Nowy plik – lista kupionych biletów

```
// src/screens/tickets/TicketsScreen.tsx – lista kupionych biletów
import { useState } from "react";
import { FlatList, Pressable, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { SCREENINGS, findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
```

```tsx
import { SegmentedControl } from "@/components/SegmentedControl";
import type { TabScreenProps } from "@/navigation/types";
import { useTickets } from "@/state/TicketsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type Filter = "upcoming" | "past";

export function TicketsScreen({ navigation }: TabScreenProps<"TicketsTab">) {
  const theme = useAppTheme();
  const { tickets } = useTickets();
  const [filter, setFilter] = useState<Filter>("upcoming");

  // Czas odczytujemy raz (leniwy inicjalizator) – render musi być „czysty”
  const [now] = useState(() => Date.now());
  const visible = tickets.filter((t) => {
    const s = SCREENINGS.find((x) => x.id === t.screeningId);
    const isPast = s ? new Date(s.startsAt).getTime() < now : false;
    return filter === "past" ? isPast : !isPast;
  });

  return (
    <View style={styles.flex}>
      <View style={styles.segmented}>
        <SegmentedControl<Filter>
          options={[
            { value: "upcoming", label: "Nadchodzące" },
            { value: "past", label: "Archiwalne" },
          ]}
          value={filter}
          onChange={setFilter}
        />
      </View>
      <FlatList
        data={visible}
        keyExtractor={(t) => t.id}
        contentContainerStyle={styles.content}
        ItemSeparatorComponent={() => <View style={styles.separator} />}
        ListEmptyComponent={<EmptyState icon="ticket-outline" title="Brak biletów" />}
        renderItem={({ item }) => {
          const screening = SCREENINGS.find((s) => s.id === item.screeningId);
          const movie = screening && findMovie(screening.movieId);
          return (
            <Pressable
              onPress={() => navigation.navigate("TicketDetails", { ticketId: item.id })}
              style={[styles.card, { backgroundColor: theme.colors.surface }]}
            >
              <Ionicons name="ticket" size={28} color={theme.colors.primary} />
              <View style={styles.flex}>
                <AppText variant="subtitle">{movie?.title}</AppText>
                {screening && (
                  <AppText variant="caption" muted>
                    {formatDayLabel(screening.startsAt)}, {formatTime(screening.startsAt)} ·
                    miejsca: {item.seats.join(", ")}
                  </AppText>
                )}
              </View>
              <AppText variant="subtitle">{formatPrice(item.total)}</AppText>
            </Pressable>
          );
        }}
      />
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  flex: { flex: 1 },
  segmented: { paddingVertical: spacing.md },
  content: { paddingHorizontal: spacing.lg, paddingBottom: spacing.xxl },
  separator: { height: spacing.sm },
  card: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

src/screens/tickets/TicketDetailsScreen.tsx

Nowy plik – szczegóły biletu + nawigacja zagnieżdżona

```
// src/screens/tickets/TicketDetailsScreen.tsx – szczegóły biletu + nawigacja zagnieżdżona
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { CommonActions } from "@react-navigation/native";
import { SCREENINGS, findHall, findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import type { RootStackScreenProps } from "@/navigation/types";
import { useTickets } from "@/state/TicketsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function TicketDetailsScreen({ navigation, route }:
RootStackScreenProps<"TicketDetails">) {
  const theme = useAppTheme();
  const { tickets } = useTickets();
  const ticket = tickets.find((t) => t.id === route.params.ticketId);
  const screening = ticket && SCREENINGS.find((s) => s.id === ticket.screeningId);
  const movie = screening && findMovie(screening.movieId);
  const hall = screening && findHall(screening.hallId);

  if (!ticket || !screening || !movie || !hall) {
    return <EmptyState icon="ticket-outline" title="Nie znaleziono biletu" />;
  }

  // Pełna ścieżka do ekranu w głęboko zagnieżdżonym nawigatorze
  const openMovie = () =>
    navigation.navigate("Main", {
      screen: "HomeTabs",
      params: {
        screen: "RepertoireTab",
        params: { screen: "MovieDetails", params: { movieId: movie.id } },
      },
    });

  // reset: nowa historia – lista biletów jako jedyny ekran pod spodem
  const goToTickets = () =>
    navigation.dispatch(
      CommonActions.reset({
        index: 0,
        routes: [
          { name: "Main", params: { screen: "HomeTabs", params: { screen: "TicketsTab" } } },
        ],
      }),
    );

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={[styles.ticket, { backgroundColor: theme.colors.surface }]}>
```

```
        <AppText variant="label" color={theme.colors.primary}>
          Bilet {ticket.id}
        </AppText>
        <AppText variant="hero">{movie.title}</AppText>
        <Row
          label="Termin"
          value={`${formatDayLabel(screening.startsAt)}, ${formatTime(screening.startsAt)}`}
        />
        <Row label="Sala" value={`${hall.name} (${hall.type})`} />
        <Row label="Miejsca" value={ticket.seats.join(", ")} />
        <Row label="Cena" value={formatPrice(ticket.total)} />
        <AppText variant="caption" muted>
          Kod QR biletu dodamy w W07.
        </AppText>
      </View>
      <Pressable style={[styles.button, { borderColor: theme.colors.primary }]}
onPress={openMovie}>
        <AppText color={theme.colors.primary}>Szczegóły filmu</AppText>
      </Pressable>
      <Pressable
        style={[styles.button, { backgroundColor: theme.colors.primary }]}
        onPress={goToTickets}
      >
        <AppText color={theme.colors.onPrimary}>Moje bilety</AppText>
      </Pressable>
    </ScrollView>
  );
}

function Row({ label, value }: { label: string; value: string }) {
  return (
    <View style={styles.row}>
      <AppText muted>{label}</AppText>
      <AppText variant="subtitle">{value}</AppText>
    </View>
  );
}

const styles = StyleSheet.create({
  content: { padding: spacing.lg, gap: spacing.md },
  ticket: { padding: spacing.xl, borderRadius: radius.lg, gap: spacing.md },
  row: { flexDirection: "row", justifyContent: "space-between", gap: spacing.md },
  button: {
    height: 50,
    borderRadius: radius.md,
    borderWidth: 1.5,
    borderColor: "transparent",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

src/screens/profile/ProfileScreen.tsx

Nowy plik – profil + przykłady nawigacji między nawigatorami

```
// src/screens/profile/ProfileScreen.tsx – profil + przykłady nawigacji między nawigatorami
import { Pressable, ScrollView, StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { DrawerActions } from "@react-navigation/native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import type { TabScreenProps } from "@/navigation/types";
import { useTickets } from "@/state/TicketsContext";
import { radius, spacing, useAppTheme } from "@/theme/theme";
```

```
type IconName = keyof typeof Ionicons.glyphMap;

export function ProfileScreen({ navigation }: TabScreenProps<"ProfileTab">) {
  const theme = useAppTheme();
  const { user, signOut } = useAuth();
  const { tickets } = useTickets();

  const items: { icon: IconName; label: string; onPress: () => void }[] = [
    // Inna zakładka tego samego nawigatora
    {
      icon: "ticket-outline",
      label: `Moje bilety (${tickets.length})`,
      onPress: () => navigation.navigate("TicketsTab"),
    },
    // Ekran nawigatora nadrzędnego (Drawer) – navigate przechodzi w górę drzewa
    {
      icon: "settings-outline",
      label: "Ustawienia",
      onPress: () => navigation.navigate("Settings"),
    },
    // Akcja nawigatora nadrzędnego wysłana przez dispatch
    {
      icon: "menu-outline",
      label: "Otwórz menu",
      onPress: () => navigation.dispatch(DrawerActions.openDrawer()),
    },
    // Zagnieżdżony ekran w innej zakładce
    {
      icon: "calendar-outline",
      label: "Jutrzejsze seanse",
      onPress: () =>
        navigation.navigate("ScreeningsTab", { screen: "Tomorrow", params: { dayOffset: 1 } }),
    },
  ];

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={styles.header}>
        <View style={[styles.avatar, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="hero" color={theme.colors.onPrimary}>
            {user?.name.charAt(0).toUpperCase()}
          </AppText>
        </View>
        <AppText variant="title">{user?.name}</AppText>
        <AppText muted>{user?.isGuest ? "Konto gościa" : user?.email}</AppText>
      </View>
      {items.map((item) => (
        <Pressable
          key={item.label}
          onPress={item.onPress}
          style={[styles.item, { backgroundColor: theme.colors.surface }]}
        >
          <Ionicons name={item.icon} size={22} color={theme.colors.primary} />
          <AppText style={styles.flex}>{item.label}</AppText>
          <Ionicons name="chevron-forward" size={18} color={theme.colors.textMuted} />
        </Pressable>
      ))}
      {/* Wylogowanie zmienia stan -> RootNavigator usuwa ekrany aplikacji */}
      <Pressable onPress={signOut} style={[styles.item, { backgroundColor: theme.colors.surface
}]}>
        <Ionicons name="log-out-outline" size={22} color={theme.colors.danger} />
        <AppText color={theme.colors.danger}>Wyloguj</AppText>
      </Pressable>
    </ScrollView>
  );
}
```

```
const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.sm },
  header: { alignItems: "center", gap: 4, marginBottom: spacing.lg },
  avatar: {
    width: 88,
    height: 88,
    borderRadius: 44,
    alignItems: "center",
    justifyContent: "center",
    marginBottom: spacing.sm,
  },
  item: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

10. Dolne zakładki

Ikony zależne od stanu (`focused`), plakietka z liczbą biletów, `backBehavior="history"` (Wstecz wraca do poprzednio odwiedzonej zakładki), `popToTopOnBlur` w zakładce z filmami. Zakładka „Bilety” dla gościa jest zablokowana: `listeners.tabPress` wywołuje `e.preventDefault()` i pokazuje komunikat.

src/navigation/HomeTabs.tsx

Nowy plik – dolne zakładki

```
// src/navigation/HomeTabs.tsx – dolne zakładki
import { Alert } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";
import { DrawerToggleButton } from "@react-navigation/drawer";
import { useTickets } from "@/state/TicketsContext";
import { useAppTheme } from "@/theme/theme";
import { RepertoireStack } from "./RepertoireStack";
import { ScreeningsTopTabs } from "./ScreeningsTopTabs";
import type { TabParamList } from "./types";
import { TicketsScreen } from "@/screens/tickets/TicketsScreen";
import { ProfileScreen } from "@/screens/profile/ProfileScreen";

const Tab = createBottomTabNavigator<TabParamList>();

type IconName = keyof typeof Ionicons.glyphMap;
const ICONS: Record<keyof TabParamList, [IconName, IconName]> = {
  RepertoireTab: ["film", "film-outline"],
  ScreeningsTab: ["calendar", "calendar-outline"],
  TicketsTab: ["ticket", "ticket-outline"],
  ProfileTab: ["person-circle", "person-circle-outline"],
};

export function HomeTabs() {
  const theme = useAppTheme();
  const { tickets } = useTickets();

  return (
    <Tab.Navigator
      backBehavior="history" // „wstecz” wraca do poprzednio odwiedzonej zakładki
      screenOptions={({ route }) => ({
```

```
        tabBarIcon: ({ focused, color, size }) => {
          const [active, inactive] = ICONS[route.name];
          return <Ionicons name={focused ? active : inactive} size={size} color={color} />;
        },
        tabBarActiveTintColor: theme.colors.primary,
        tabBarInactiveTintColor: theme.colors.textMuted,
        tabBarHideOnKeyboard: true, // Android: pasek znika, gdy pojawia się klawiatura
        animation: "shift", // animacja przy przełączaniu zakładek
        headerLeft: () => <DrawerToggleButton tintColor={theme.colors.primary} />,
      })}
      // Nasłuchiwanie zdarzeń wszystkich zakładek
      screenListeners={{
        tabPress: (e) => console.log("[zakładka]", e.target),
      }}
    >
      <Tab.Screen
        name="RepertoireTab"
        component={RepertoireStack}
        options={{
          title: "Repertuar",
          headerShown: false, // nagłówek pochodzi z zagnieżdżonego Stacka
          popToTopOnBlur: true, // po opuszczeniu zakładki wraca do listy filmów
        }}
      />
      <Tab.Screen
        name="ScreeningsTab"
        component={ScreeningsTopTabs}
        options={{ title: "Seanse" }}
      />
      <Tab.Screen
        name="TicketsTab"
        component={TicketsScreen}
        options={{
          title: "Bilety",
          tabBarBadge: tickets.length > 0 ? tickets.length : undefined,
        }}
        // Zdarzenie tylko dla tej zakładki – można je anulować
        listeners={{
          tabPress: (e) => {
            if (tickets.length === 0) {
              e.preventDefault();
              Alert.alert("Brak biletów", "Kup pierwszy bilet w zakładce Seanse.");
            }
          },
        }}
      />
      <Tab.Screen name="ProfileTab" component={ProfileScreen} options={{ title: "Profil" }} />
    </Tab.Navigator>
  );
}
```

11. Menu boczne (Drawer)

Własna zawartość menu: nagłówek z użytkownikiem, standardowa lista ekranów (`DrawerItemList`) i dodatkowy element „Wyloguj” (`DrawerItem`). Ekrany menu: lista sal, ustawienia, informacje o aplikacji.

src/navigation/MainDrawer.tsx

Nowy plik – menu boczne z własną zawartością

```
// src/navigation/MainDrawer.tsx – menu boczne z własną zawartością
import { StyleSheet, View } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import {
  DrawerContentScrollView,
```

```
  DrawerItem,
  DrawerItemList,
  createDrawerNavigator,
  type DrawerContentComponentProps,
} from "@react-navigation/drawer";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import { spacing, useAppTheme } from "@/theme/theme";
import { HomeTabs } from "./HomeTabs";
import type { DrawerParamList } from "./types";
import { HallsScreen } from "@/screens/drawer/HallsScreen";
import { SettingsScreen } from "@/screens/drawer/SettingsScreen";
import { AboutScreen } from "@/screens/drawer/AboutScreen";

const Drawer = createDrawerNavigator<DrawerParamList>();

function DrawerContent(props: DrawerContentComponentProps) {
  const theme = useAppTheme();
  const { user, signOut } = useAuth();
  return (
    <DrawerContentScrollView {...props}>
      <View style={styles.header}>
        <View style={[styles.avatar, { backgroundColor: theme.colors.primary }]}>
          <AppText variant="title" color={theme.colors.onPrimary}>
            {user?.name.charAt(0).toUpperCase() ?? "?"}
          </AppText>
        </View>
        <AppText variant="subtitle">{user?.name}</AppText>
        <AppText variant="caption" muted>
          {user?.isGuest ? "Gość" : user?.email}
        </AppText>
      </View>
      {/* Standardowe pozycje generowane z ekranów nawigatora */}
      <DrawerItemList {...props} />
      <DrawerItem
        label="Wyloguj"
        icon={({ size }) => (
          <Ionicons name="log-out-outline" size={size} color={theme.colors.danger} />
        )}
        labelStyle={{ color: theme.colors.danger }}
        onPress={signOut}
      />
    </DrawerContentScrollView>
  );
}

export function MainDrawer() {
  const theme = useAppTheme();
  return (
    <Drawer.Navigator
      drawerContent={(props) => <DrawerContent {...props} />}
      screenOptions={{
        drawerType: "front", // „slide” przesuwa też ekran, „permanent” – stałe menu (tablet)
        drawerActiveTintColor: theme.colors.primary,
        drawerActiveBackgroundColor: theme.dark ? "#2A1215" : "#FDECEC",
        swipeEdgeWidth: 40, // obszar przy krawędzi, z którego można wysunąć menu gestem
        headerTintColor: theme.colors.primary,
      }}
    >
      <Drawer.Screen
        name="HomeTabs"
        component={HomeTabs}
        options={{
          title: "Kino",
          headerShown: false, // nagłówki pokazują zakładki i ich stacki
          drawerIcon: ({ color, size }) => (
            <Ionicons name="home-outline" size={size} color={color} />
```

```
          ),
        }}
      />
      <Drawer.Screen
        name="Halls"
        component={HallsScreen}
        options={{
          title: "Sale kinowe",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="grid-outline" size={size} color={color} />
          ),
        }}
      />
      <Drawer.Screen
        name="Settings"
        component={SettingsScreen}
        options={{
          title: "Ustawienia",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="settings-outline" size={size} color={color} />
          ),
        }}
      />
      <Drawer.Screen
        name="About"
        component={AboutScreen}
        options={{
          title: "O aplikacji",
          drawerIcon: ({ color, size }) => (
            <Ionicons name="information-circle-outline" size={size} color={color} />
          ),
        }}
      />
    </Drawer.Navigator>
  );
}

const styles = StyleSheet.create({
  header: { padding: spacing.lg, gap: 4 },
  avatar: {
    width: 56,
    height: 56,
    borderRadius: 28,
    alignItems: "center",
    justifyContent: "center",
    marginBottom: spacing.sm,
  },
});
```

src/screens/drawer/HallsScreen.tsx

Nowy plik – lista sal (ekran menu bocznego)

```
// src/screens/drawer/HallsScreen.tsx – lista sal (ekran menu bocznego)
import { FlatList, StyleSheet, View } from "react-native";
import { MaterialCommunityIcons } from "@expo/vector-icons";
import { HALLS } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function HallsScreen() {
  const theme = useAppTheme();
  return (
    <FlatList
      data={HALLS}
      keyExtractor={(h) => h.id}
      contentContainerStyle={styles.content}
```

```
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      renderItem={({ item }) => (
        <View style={[styles.card, { backgroundColor: theme.colors.surface }]}>
          <MaterialCommunityIcons name="theater" size={32} color={theme.colors.primary} />
          <View style={styles.flex}>
            <AppText variant="subtitle">{item.name}</AppText>
            <AppText variant="caption" muted>
              {item.type} · {item.rows} rzędów · {item.rows * item.seatsPerRow} miejsc
            </AppText>
          </View>
        </View>
      )}
    />
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg },
  separator: { height: spacing.sm },
  card: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

src/screens/drawer/SettingsScreen.tsx

Nowy plik – ustawienia (działające opcje dodamy w W04)

```
// src/screens/drawer/SettingsScreen.tsx – ustawienia (działające opcje dodamy w W04)
import { useState } from "react";
import { ScrollView, StyleSheet, Switch, View } from "react-native";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function SettingsScreen() {
  const theme = useAppTheme();
  const [reminders, setReminders] = useState(true);
  const [newsletter, setNewsletter] = useState(false);

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <View style={[styles.card, { backgroundColor: theme.colors.surface }]}>
        <Row label="Przypomnienia o seansach" value={reminders} onChange={setReminders} />
        <Row label="Newsletter z premierami" value={newsletter} onChange={setNewsletter} />
      </View>
      <AppText variant="caption" muted>
        Motyw, ulubione i trwały zapis ustawień – w W04.
      </AppText>
    </ScrollView>
  );
}

function Row({
  label,
  value,
  onChange,
}: {
  label: string;
  value: boolean;
  onChange: (v: boolean) => void;
}) {
```

```
  return (
    <View style={styles.row}>
      <AppText style={styles.flex}>{label}</AppText>
      <Switch value={value} onValueChange={onChange} />
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { padding: spacing.lg, gap: spacing.md },
  card: { borderRadius: radius.md, paddingHorizontal: spacing.lg },
  row: { flexDirection: "row", alignItems: "center", minHeight: 56, gap: spacing.md },
});
```

src/screens/drawer/AboutScreen.tsx

Nowy plik – informacje + linki wychodzące (Linking)

```
// src/screens/drawer/AboutScreen.tsx – informacje + linki wychodzące (Linking)
import { Alert, Linking, Pressable, ScrollView, StyleSheet } from "react-native";
import { Ionicons } from "@expo/vector-icons";
import * as ExpoLinking from "expo-linking";
import { AppText } from "@/components/AppText";
import { radius, spacing, useAppTheme } from "@/theme/theme";

type IconName = keyof typeof Ionicons.glyphMap;

export function AboutScreen() {
  const theme = useAppTheme();

  // Linking.openURL – otwarcie innej aplikacji (przeglądarka, telefon, poczta)
  const open = async (url: string) => {
    const supported = await Linking.canOpenURL(url);
    if (supported) await Linking.openURL(url);
    else Alert.alert("Nie można otworzyć", url);
  };

  const links: { icon: IconName; label: string; url: string }[] = [
    { icon: "globe-outline", label: "Strona kina", url: "https://example.com" },
    { icon: "call-outline", label: "Zadzwoń do kasy", url: "tel:+48123456789" },
    { icon: "mail-outline", label: "Napisz do nas", url: "mailto:kontakt@example.com" },
  ];

  return (
    <ScrollView contentContainerStyle={styles.content}>
      <AppText variant="title">Kino Pod Gwiazdami</AppText>
      <AppText muted>Aplikacja przykładowa do wykładu „Aplikacje mobilne”.</AppText>
      {links.map((l) => (
        <Pressable
          key={l.url}
          onPress={() => open(l.url)}
          style={[styles.item, { backgroundColor: theme.colors.surface }]}
        >
          <Ionicons name={l.icon} size={22} color={theme.colors.primary} />
          <AppText>{l.label}</AppText>
        </Pressable>
      ))}
      <AppText variant="caption" muted selectable>
        Deep link do tego ekranu: {ExpoLinking.createURL("about")}
      </AppText>
    </ScrollView>
  );
}
```

```
const styles = StyleSheet.create({
  content: { padding: spacing.lg, gap: spacing.md },
  item: {
    flexDirection: "row",
    alignItems: "center",
    gap: spacing.md,
    padding: spacing.lg,
    borderRadius: radius.md,
  },
});
```

12. Logowanie, wybór miejsc i ekran 404

Ekrany niezalogowanego użytkownika (powitanie, uproszczone logowanie – pełny formularz na wykładzie 5). `SeatSelectionScreen` to modal z blokadą wyjścia: gdy wybrano miejsca, `usePreventRemove` przechwytuje próbę zamknięcia (gest, przycisk, Wstecz) i pyta o potwierdzenie. Zakup wykonuje `navigation.replace` – ekran biletu zastępuje wybór miejsc, więc „Wstecz" nie wraca do kupionego wyboru.

src/screens/auth/WelcomeScreen.tsx

Nowy plik – ekran powitalny (tylko dla niezalogowanych)

```
// src/screens/auth/WelcomeScreen.tsx – ekran powitalny (tylko dla niezalogowanych)
import { Pressable, StyleSheet, View } from "react-native";
import { Image } from "expo-image";
import { LinearGradient } from "expo-linear-gradient";
import { SafeAreaView } from "react-native-safe-area-context";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import { MOVIES } from "@/data/mockData";
import type { RootStackScreenProps } from "@/navigation/types";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function WelcomeScreen({ navigation }: RootStackScreenProps<"Welcome">) {
  const theme = useAppTheme();
  const { signIn } = useAuth();

  return (
    <View style={styles.flex}>
      <Image source={MOVIES[1].backdropUrl} style={StyleSheet.absoluteFill} contentFit="cover"
/>
      <LinearGradient colors={["rgba(0,0,0,0.2)", "#000000"]} style={StyleSheet.absoluteFill} />
      <SafeAreaView style={[styles.flex, styles.content]}>
        <AppText variant="label" color={theme.colors.primary}>
          Kino Pod Gwiazdami
        </AppText>
        <AppText variant="hero" color="#FFFFFF">
          Twoje kino w kieszeni
        </AppText>
        <AppText color="#D1D1D6">
          Repertuar, rezerwacje miejsc i bilety – wszystko w jednym miejscu.
        </AppText>
        <Pressable
          style={[styles.button, { backgroundColor: theme.colors.primary }]}
          onPress={() => navigation.navigate("SignIn")}
        >
          <AppText variant="subtitle" color={theme.colors.onPrimary}>
            Zaloguj się
          </AppText>
        </Pressable>
        {/* Zmiana stanu logowania sama przełączy zestaw ekranów w RootNavigator */}
        <Pressable
          style={[styles.button, styles.ghost]}
          onPress={() => signIn({ name: "Gość", isGuest: true })}
```

```
        >
          <AppText variant="subtitle" color="#FFFFFF">
            Kontynuuj jako gość
          </AppText>
        </Pressable>
      </SafeAreaView>
    </View>
  );
}

const styles = StyleSheet.create({
  flex: { flex: 1 },
  content: { justifyContent: "flex-end", padding: spacing.xl, gap: spacing.md },
  button: {
    height: 52,
    borderRadius: radius.md,
    alignItems: "center",
    justifyContent: "center",
  },
  ghost: { borderWidth: 1, borderColor: "rgba(255,255,255,0.4)" },
});
```

src/screens/auth/SignInScreen.tsx

Nowy plik – uproszczone logowanie (pełny formularz w W05)

```
// src/screens/auth/SignInScreen.tsx – uproszczone logowanie (pełny formularz w W05)
import { useState } from "react";
import { Pressable, StyleSheet, TextInput, View } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import type { RootStackScreenProps } from "@/navigation/types";
import { fonts, radius, spacing, useAppTheme } from "@/theme/theme";

export function SignInScreen({ route }: RootStackScreenProps<"SignIn">) {
  const theme = useAppTheme();
  const { signIn } = useAuth();
  // Parametr opcjonalny (np. z deep linku kino://login?email=...)
  const [email, setEmail] = useState(route.params?.email ?? "");
  const canSubmit = email.includes("@");

  return (
    <View style={styles.container}>
      <AppText variant="title">Witaj ponownie</AppText>
      <TextInput
        value={email}
        onChangeText={setEmail}
        placeholder="E-mail"
        placeholderTextColor={theme.colors.textMuted}
        keyboardType="email-address"
        autoCapitalize="none"
        style={[
          styles.input,
          {
            color: theme.colors.text,
            borderColor: theme.colors.border,
            backgroundColor: theme.colors.surface,
          },
        ]}
      />
      <Pressable
        disabled={!canSubmit}
        onPress={() => signIn({ name: email.split("@")[0], email, isGuest: false })}
        style={[styles.button, { backgroundColor: theme.colors.primary }, !canSubmit &&
styles.dim]}
      >
```

```
        <AppText variant="subtitle" color={theme.colors.onPrimary}>
          Zaloguj
        </AppText>
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: spacing.xl, gap: spacing.lg },
  input: {
    height: 50,
    borderWidth: 1,
    borderRadius: radius.md,
    paddingHorizontal: spacing.md,
    fontFamily: fonts.regular,
    fontSize: 16,
  },
  button: { height: 50, borderRadius: radius.md, alignItems: "center", justifyContent: "center"
},
  dim: { opacity: 0.5 },
});
```

src/screens/booking/SeatSelectionScreen.tsx

Nowy plik – wybór miejsc (modal) z blokadą wyjścia

```
// src/screens/booking/SeatSelectionScreen.tsx – wybór miejsc (modal) z blokadą wyjścia
import { useState } from "react";
import { Alert, Pressable, ScrollView, StyleSheet, View, useWindowDimensions } from "react-
native";
import { usePreventRemove } from "@react-navigation/native";
import { useSafeAreaInsets } from "react-native-safe-area-context";
import { SCREENINGS, findHall, findMovie } from "@/data/mockData";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import { SeatMap } from "@/components/SeatMap";
import type { RootStackScreenProps } from "@/navigation/types";
import { useTickets } from "@/state/TicketsContext";
import { formatDayLabel, formatPrice, formatTime } from "@/utils/format";
import { radius, spacing, useAppTheme } from "@/theme/theme";

export function SeatSelectionScreen({ navigation, route }:
RootStackScreenProps<"SeatSelection">) {
  const theme = useAppTheme();
  const insets = useSafeAreaInsets();
  const { width } = useWindowDimensions();
  const { addTicket } = useTickets();
  const [selected, setSelected] = useState<string[]>([]);

  // Zapobiegamy przypadkowemu zamknięciu ekranu z niezapisanym wyborem
  usePreventRemove(selected.length > 0, ({ data }) => {
    // Nasze własne „replace” po zakupie przepuszczamy bez pytania (stan nie zdążyłby się
zmienić)
    if (data.action.type === "REPLACE") {
      navigation.dispatch(data.action);
      return;
    }
    Alert.alert("Porzucić wybór?", "Wybrane miejsca nie zostaną zarezerwowane.", [
      { text: "Zostań", style: "cancel" },
      // Wykonujemy akcję nawigacji, którą wcześniej zablokowaliśmy
      { text: "Porzuć", style: "destructive", onPress: () => navigation.dispatch(data.action) },
    ]);
  });

  const screening = SCREENINGS.find((s) => s.id === route.params.screeningId);
```

```
  const hall = screening && findHall(screening.hallId);
  const movie = screening && findMovie(screening.movieId);
  if (!screening || !hall || !movie) {
    return <EmptyState title="Seans nie istnieje" message="Sprawdź link lub wybierz inny seans."
/>;
  }

  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]));

  const confirm = () => {
    const ticket = addTicket({
      screeningId: screening.id,
      seats: selected,
      total: selected.length * screening.price,
    });
    // replace: modal wyboru miejsc zostaje zastąpiony biletem – „wstecz” nie wraca do wyboru
    navigation.replace("TicketDetails", { ticketId: ticket.id });
  };

  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}
        </AppText>
        <ScrollView horizontal contentContainerStyle={styles.mapScroll}>
          <SeatMap
            hall={hall}
            taken={screening.takenSeats}
            selected={selected}
            onToggle={toggleSeat}
            seatSize={seatSize}
          />
        </ScrollView>
      </ScrollView>
      <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(", ")}` : "Wybierz miejsca"}
          </AppText>
          <AppText variant="subtitle">{formatPrice(selected.length * screening.price)}</AppText>
        </View>
        <Pressable
          disabled={selected.length === 0}
          onPress={confirm}
          style={[
            styles.button,
            { backgroundColor: theme.colors.primary },
            selected.length === 0 && styles.dim,
          ]}
        >
          <AppText variant="subtitle" color={theme.colors.onPrimary}>
            Kupuję
          </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
},
  dim: { opacity: 0.5 },
});
```

src/screens/NotFoundScreen.tsx

Nowy plik – nieznany deep link

```
// src/screens/NotFoundScreen.tsx – nieznany deep link
import { Pressable, StyleSheet, View } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { AppText } from "@/components/AppText";
import { EmptyState } from "@/components/EmptyState";
import type { RootStackScreenProps } from "@/navigation/types";
import { spacing, useAppTheme } from "@/theme/theme";

export function NotFoundScreen({ navigation }: RootStackScreenProps<"NotFound">) {
  const theme = useAppTheme();
  const { user } = useAuth();
  // Dostępne ekrany zależą od stanu logowania (guard w RootNavigator)
  const goHome = () => (user ? navigation.navigate("Main") : navigation.navigate("Welcome"));
  return (
    <View style={styles.container}>
      <EmptyState
        icon="help-circle-outline"
        title="Nie ma takiej strony"
        message="Link jest nieprawidłowy."
      />
      <Pressable onPress={() => (navigation.canGoBack() ? navigation.goBack() : goHome())}>
        <AppText color={theme.colors.primary}>Wróć</AppText>
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, alignItems: "center", justifyContent: "center", gap: spacing.lg },
});
```

13. Deep linki

Konfiguracja `linking` odwzorowuje zagnieżdżenie nawigatorów. Parametry z adresu są zawsze tekstem – `parse` zamienia je na liczby/tablice, a `stringify` z powrotem. Ścieżka `*` łapie nieznane adresy (ekran „Nie znaleziono”).

src/navigation/linking.ts

Nowy plik – deep linki: adres URL -> ekran (z parametrami)

```
// src/navigation/linking.ts – deep linki: adres URL -> ekran (z parametrami)
import * as Linking from "expo-linking";
import type { LinkingOptions } from "@react-navigation/native";
import type { RootStackParamList } from "./types";

export const linking: LinkingOptions<RootStackParamList> = {
  // createURL("/") w Expo Go zwraca exp://IP:8081/--/, w buildzie: kino://
  prefixes: [Linking.createURL("/"), "kino://", "https://kino.example.com"],
  config: {
    // Struktura odpowiada zagnieżdżeniu nawigatorów
    screens: {
      Welcome: "welcome",
      SignIn: "login",
      Main: {
        screens: {
          HomeTabs: {
            screens: {
              RepertoireTab: {
                screens: {
                  Movies: "movies",
                  MovieDetails: "movie/:movieId",
                  Filters: "filters",
                },
              },
              ScreeningsTab: {
                screens: {
                  Today: "screenings/today",
                  Tomorrow: "screenings/tomorrow",
                  DayAfter: "screenings/day-after",
                },
              },
              TicketsTab: "tickets",
              ProfileTab: "profile",
            },
          },
          Halls: "halls",
          Settings: "settings",
          About: "about",
        },
      },
      SeatSelection: "booking/:screeningId",
      TicketDetails: "ticket/:ticketId",
      Trailer: "trailer/:movieId",
      NotFound: "*", // każdy nieznany adres
    },
  },
};
```

14. Root Stack z ochroną tras

Najważniejszy plik etapu. Gdy `user === null`, istnieją tylko ekrany powitania i logowania – nie da się „wejść" do aplikacji nawet deep linkiem. Po zalogowaniu React Navigation sam przełącza zestaw ekranów. Grupa `Stack.Group` z `presentation: "modal"` nadaje wspólne opcje kilku ekranom. `onStateChange` loguje przejścia między ekranami (na wykładzie 9 zastąpi to analityka).

src/navigation/RootNavigator.tsx

Nowy plik – kontener nawigacji + główny Stack z ochroną tras

```
// src/navigation/RootNavigator.tsx – kontener nawigacji + główny Stack z ochroną tras
import { useRef } from "react";
```

```
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { ActivityIndicator } from "react-native";
import { useAuth } from "@/auth/AuthContext";
import { useAppTheme } from "@/theme/theme";
import { linking } from "./linking";
import { navigationRef } from "./navigationRef";
import { toNavigationTheme } from "./navTheme";
import { MainDrawer } from "./MainDrawer";
import type { RootStackParamList } from "./types";
import { WelcomeScreen } from "@/screens/auth/WelcomeScreen";
import { SignInScreen } from "@/screens/auth/SignInScreen";
import { SeatSelectionScreen } from "@/screens/booking/SeatSelectionScreen";
import { TrailerScreen } from "@/screens/movies/TrailerScreen";
import { TicketDetailsScreen } from "@/screens/tickets/TicketDetailsScreen";
import { NotFoundScreen } from "@/screens/NotFoundScreen";

const Stack = createNativeStackNavigator<RootStackParamList>();

export function RootNavigator() {
  const theme = useAppTheme();
  const { user } = useAuth();
  const routeNameRef = useRef<string | undefined>(undefined);

  return (
    <NavigationContainer
      ref={navigationRef}
      theme={toNavigationTheme(theme)}
      linking={linking}
      fallback={<ActivityIndicator style={{ flex: 1 }} />} // podczas obsługi deep linku
      onReady={() => {
        routeNameRef.current = navigationRef.getCurrentRoute()?.name;
      }}
      // Śledzenie ekranów – podstawa analityki (rozwijamy w W09)
      onStateChange={() => {
        const current = navigationRef.getCurrentRoute()?.name;
        if (current && current !== routeNameRef.current) {
          console.log(`[nawigacja] ${routeNameRef.current} -> ${current}`);
        }
        routeNameRef.current = current;
      }}
    >
      <Stack.Navigator
        screenOptions={{
          headerTintColor: theme.colors.primary,
          headerBackButtonDisplayMode: "minimal", // iOS: sama strzałka, bez tekstu
          contentStyle: { backgroundColor: theme.colors.background },
        }}
      >
        {user === null ? (
          // Niezalogowany: istnieją tylko ekrany logowania (guard przez renderowanie warunkowe)
          <Stack.Group screenOptions={{ headerShown: false, animation: "fade" }}>
            <Stack.Screen name="Welcome" component={WelcomeScreen} />
            <Stack.Screen
              name="SignIn"
              component={SignInScreen}
              options={{ headerShown: true, title: "Logowanie", animation: "slide_from_right" }}
            />
          </Stack.Group>
        ) : (
          <>
            <Stack.Screen name="Main" component={MainDrawer} options={{ headerShown: false }} />
            {/* Grupa modalna: ekran nasuwa się od dołu */}
            <Stack.Group screenOptions={{ presentation: "modal" }}>
              <Stack.Screen
                name="SeatSelection"
                component={SeatSelectionScreen}
```

```
                options={{ title: "Wybór miejsc", gestureEnabled: false }}
              />
            </Stack.Group>
            <Stack.Screen
              name="TicketDetails"
              component={TicketDetailsScreen}
              // Tytuł zależny od parametru trasy
              options={({ route }) => ({ title: `Bilet ${route.params.ticketId}` })}
            />
            {/* Przezroczysty modal: poprzedni ekran prześwituje pod spodem */}
            <Stack.Screen
              name="Trailer"
              component={TrailerScreen}
              options={{ presentation: "transparentModal", animation: "fade", headerShown: false
}}
            />
          </>
        )}
        <Stack.Screen
          name="NotFound"
          component={NotFoundScreen}
          options={{ title: "Nie znaleziono" }}
        />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

15. App.tsx – providery

`GestureHandlerRootView` (gesty Drawera) → `SafeAreaProvider` → providery stanu → `RootNavigator`.

App.tsx

Zastąp całą zawartość pliku – etap W03: providery + nawigacja

```
// App.tsx – etap W03: providery + nawigacja
import { useEffect } from "react";
import { StatusBar } from "expo-status-bar";
import * as SplashScreen from "expo-splash-screen";
import { GestureHandlerRootView } from "react-native-gesture-handler";
import { SafeAreaProvider } from "react-native-safe-area-context";
import {
  Inter_400Regular,
  Inter_500Medium,
  Inter_600SemiBold,
  Inter_700Bold,
  Inter_900Black,
  useFonts,
} from "@expo-google-fonts/inter";
import { AuthProvider } from "@/auth/AuthContext";
import { RootNavigator } from "@/navigation/RootNavigator";
import { TicketsProvider } from "@/state/TicketsContext";

SplashScreen.preventAutoHideAsync();

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 (
    // GestureHandlerRootView – wymagany przez Drawer (gesty wysuwania menu)
    <GestureHandlerRootView style={{ flex: 1 }}>
      <SafeAreaProvider>
        <AuthProvider>
          <TicketsProvider>
            <StatusBar style="auto" />
            <RootNavigator />
          </TicketsProvider>
        </AuthProvider>
      </SafeAreaProvider>
    </GestureHandlerRootView>
  );
}
```

16. Usuń niepotrzebne pliki

Poniższe pliki z poprzedniego etapu nie są już używane – usuń je:

- `src/components/MovieDetailsModal.tsx`
- `src/screens/HallView.tsx`
- `src/screens/RepertoireView.tsx`
- `src/screens/ScreeningsView.tsx`

Terminal

```
rm src/components/MovieDetailsModal.tsx src/screens/HallView.tsx src/screens/RepertoireView.tsx
src/screens/ScreeningsView.tsx
```

Uruchomienie i sprawdzenie

Terminal

```
npx expo start -c
```

- ☐ jako gość: zakładka „Bilety” pokazuje komunikat zamiast się otworzyć,
- ☐ repertuar: wyszukiwarka w nagłówku, filtry jako arkusz (przeciągnij uchwyt – dwie wysokości), wynik filtrów wraca do listy,
- ☐ szczegóły filmu: przezroczysty nagłówek, zwiastun jako przezroczysty modal,
- ☐ wybór miejsc: zaznacz miejsce i spróbuj zamknąć modal gestem – pojawi się pytanie,
- ☐ przesuń dni w „Seansach” gestem; otwórz menu boczne gestem od lewej krawędzi,
- ☐ deep link: w terminalu wpisz polecenie poniżej (Android) – aplikacja otworzy szczegóły filmu.

Test deep linku (emulator Androida)

```
npx uri-scheme open "exp://127.0.0.1:8081/--/movie/m1" --android
```

Na co uważać

preload + setOptions

Ekran wczytany przez `navigation.preload()` nie może ustawiać przycisków nagłówka przez `setOptions` w efekcie (zostałyby ustawione dla niewidocznego ekranu i mogło to powodować błąd). Przyciski nagłówka przekazujemy w `options` nawigatora (`MovieHeaderActions`).

usePreventRemove a replace/reset

Blokada wyjścia przechwytuje KAŻDE usunięcie ekranu, także `navigation.replace` po zakupie. W callbacku przepuszczamy akcję `REPLACE` (`navigation.dispatch(data.action)`) bez pytania.

Parametry w adresie to napisy

Bez `parse`/`stringify` tablica w parametrze trafi do adresu jako `[object Object]`. Parametry przekazuj jako proste, serializowalne wartości (bez funkcji i obiektów klas).

Zagnieżdżone nagłówki

Gdy zakładka zawiera Stack, ukryj nagłówek zakładki (`headerShown: false`) – inaczej zobaczysz dwa nagłówki jeden nad drugim.

formSheet i transparentModal na Androidzie/web

Arkusz z wysokościami wygląda natywnie na iOS i Androidzie; w przeglądarce jest zwykłym modalem. Zawsze sprawdzaj nawigację na obu platformach mobilnych.

Lista kontrolna

- ☐ `npx tsc --noEmit` nie zgłasza błędów (wszystkie wywołania `navigate` są otypowane),
- ☐ nie da się otworzyć ekranów aplikacji bez logowania (także przez deep link),
- ☐ „Wstecz” na Androidzie zachowuje się sensownie w każdym nawigatorze,
- ☐ nieznany adres pokazuje ekran „Nie znaleziono”.

Zadania dodatkowe

1. Dodaj ekran „Kup ponownie” otwierany z biletu przez `navigation.push` (dwa ekrany tego samego typu na stosie).
2. Zmień `drawerType` na `"permanent"` dla szerokich ekranów (tablet) – użyj `useWindowDimensions`.
3. Dodaj deep link do ekranu filtrów z gotowym wyborem gatunków (`kino://filters?genres=Dramat,Komedia`) – użyj `parse`/`stringify` jak w `linking.ts`.