Лекція-практикум 2

Додаємо навігацію

У цій лекції ми навчимося основам Expo Router: створимо стекову навігацію та нижню панель із двома вкладками.

Основи Expo Router

Expo Router — це фреймворк на основі файлової маршрутизації для React Native і веб-застосунків. Він керує навігацією між екранами та використовує однакові компоненти для всіх платформ. Щоб почати, нам потрібно знати такі правила:

- Каталог **app**: спеціальна папка, яка містить лише маршрути та їх лейаути. Будь-які файли, додані до цього каталогу, стають екраном у нашому нативному застосунку і сторінкою в веб.
- Кореневий лейаут: файл `app/_layout.tsx`. Визначає елементи інтерфейсу, які спільні для різних маршрутів — наприклад заголовки чи таб-бар.
- Конвенції імен файлів: файли з назвою **index** (наприклад `index.tsx`) відповідають батьківському каталогу без додавання сегмента шляху. Наприклад, `app/index.tsx` відповідає маршруту `/`.
- Файли маршруту експортують React-компонент як значення за замовчуванням. Файли можуть мати розширення `.js`, `.jsx`, `.ts`, `.tsx`.
- Android, iOS і веб використовують одну і ту ж структуру навігації.

Додаємо новий екран у стек

Створімо новий файл `about.tsx` в каталозі `app`. Цей екран має показувати назву екрану, коли користувач перейде за маршрутом `/about`.

`app/about.tsx`:

```
import { Text, View, StyleSheet } from 'react-native';

export default function AboutScreen() {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>About screen</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    justifyContent: 'center',
    alignItems: 'center',
  },
  text: {
    color: '#fff',
  },
});
```

У файлі `app/_layout.tsx`:

1. Додайте `<Stack.Screen />` компонент і проп `options`, щоб оновити заголовок маршруту `/about`.
2. Оновіть заголовок для маршруту `/index` на `Home`, додавши `options` проп.

app/_layout.tsx

```
import { Stack } from 'expo-router';

export default function RootLayout() {
  return (
    <Stack>
      <Stack.Screen name="index" options={{ title: 'Home' }} />
      <Stack.Screen name="about" options={{ title: 'About' }} />
    </Stack>
  );
}
```

Що таке `Stack`?
Stack navigator — це основа навігації між різними екранами в застосунку.

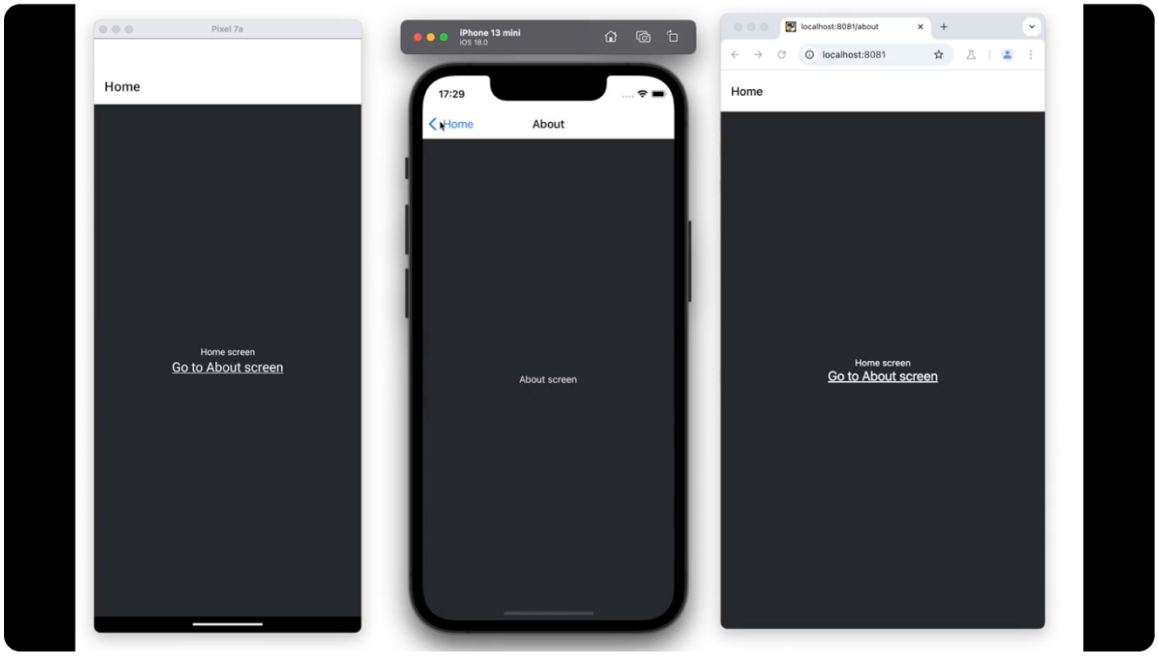
На Android маршрут у стеку анімується поверх поточного екрану. На iOS маршрут анімується справа. Expo Router надає компонент `Stack` для створення такої навігації.

Навігація між екранами

Використаємо компонент `Link` з `expo-router`, щоб перейти з маршруту `/index` до `/about`. Це React-компонент, який рендерить `<Text>` із пропом `href`.

Ось як має виглядати `app/index.tsx`:

```
import { Text, View, StyleSheet } from 'react-native';
import { Link } from 'expo-router';

export default function Index() {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Home screen</Text>
      <Link href="/about" style={styles.button}>
        Go to About screen
      </Link>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
    justifyContent: 'center',
  },
  text: {
    color: '#fff',
  },
  button: {
    fontSize: 20,
    textDecorationLine: 'underline',
    color: '#fff',
  },
});
```



Додати маршрут not-found

Коли маршрут не існує, можна використати маршрут `+not-found`, щоб показати екран заповнення (fallback). Це корисно, коли користувач вводить неправильний шлях: замість аварії застосунку або стандартної помилки веб, ви показуєте власний екран. Expo Router використовує спеціальний файл `+not-found.tsx` для цього.

1. Створіть новий файл `+not-found.tsx` всередині каталогу `app`, щоб додати компонент `NotFoundScreen`.
2. Додайте проп `options` через `Stack.Screen`, щоб встановити власний заголовок цього маршруту.
3. Додайте компонент `Link`, щоб перенаправляти назад до `/` маршруту (головної).

```tsx
import { View, StyleSheet } from 'react-native';
import { Link, Stack } from 'expo-router';

export default function NotFoundScreen() {
  return (
    <>
      <Stack.Screen options={{ title: 'Oops! Not Found' }} />
      <View style={styles.container}>
        <Link href="/" style={styles.button}>
          Go back to Home screen!
        </Link>
      </View>
    </>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    justifyContent: 'center',
    alignItems: 'center',
  },

  button: {
    fontSize: 20,
    textDecorationLine: 'underline',
    color: '#fff',
  },
});
```

Щоб протестувати це: зайдіть у браузері на URL типу `http://localhost:8081/123` — застосунок має показати екран `NotFoundScreen`.

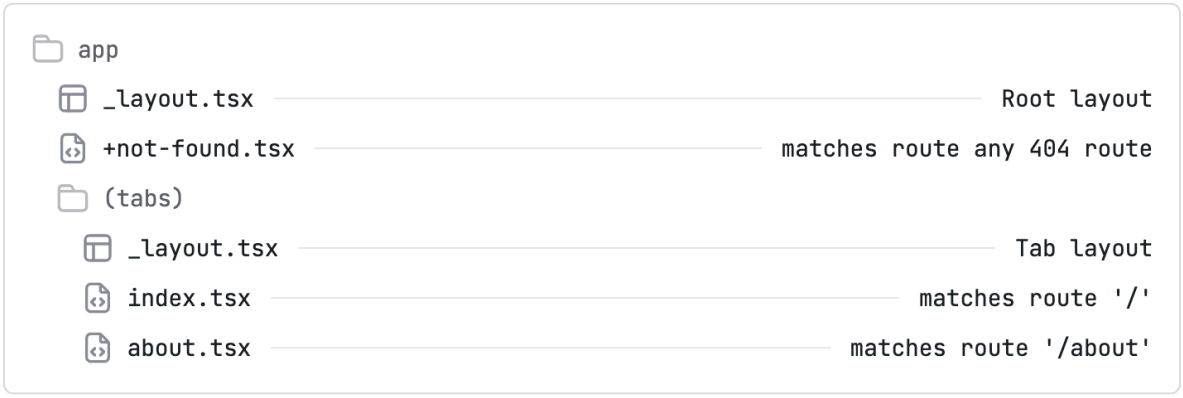
Додаємо нижню таб-навігацію (Bottom Tab Navigator)

На даному етапі структура вашої папки `app` виглядає приблизно так:

```
app
  _layout.tsx ———— Root layout
  index.tsx ———— matches route '/'
  about.tsx ———— matches route '/about'
  +not-found.tsx ———— matches route any 404 route
```

Ми додамо нижню панель табів і використаємо наявні екрани Home та About, щоб створити таб-лэйаут (частий патерн у соціальних мережах і мобільних застосунках). Також використаємо стек-навігатор у кореневому лейауті, щоб маршрут `+not-found` відображався поверх будь-яких вкладених навігаторів.

1. Всередині каталогу `app` додайте піддиректорію `(tabs)`. Ця спеціальна директорія використовуються, щоб згрупувати маршрути і відображати їх у нижньому таб-барі.
2. Створіть файл `(tabs)/_layout.tsx` — він буде визначати лейаут для табів, окремо від кореневого.
3. Перемістіть існуючі файли `index.tsx` і `about.tsx` у директорію `(tabs)`. Структура папки `app` повинна виглядати так:



Оновіть кореневий лейаут файл, щоб додати маршрут `(tabs)`:

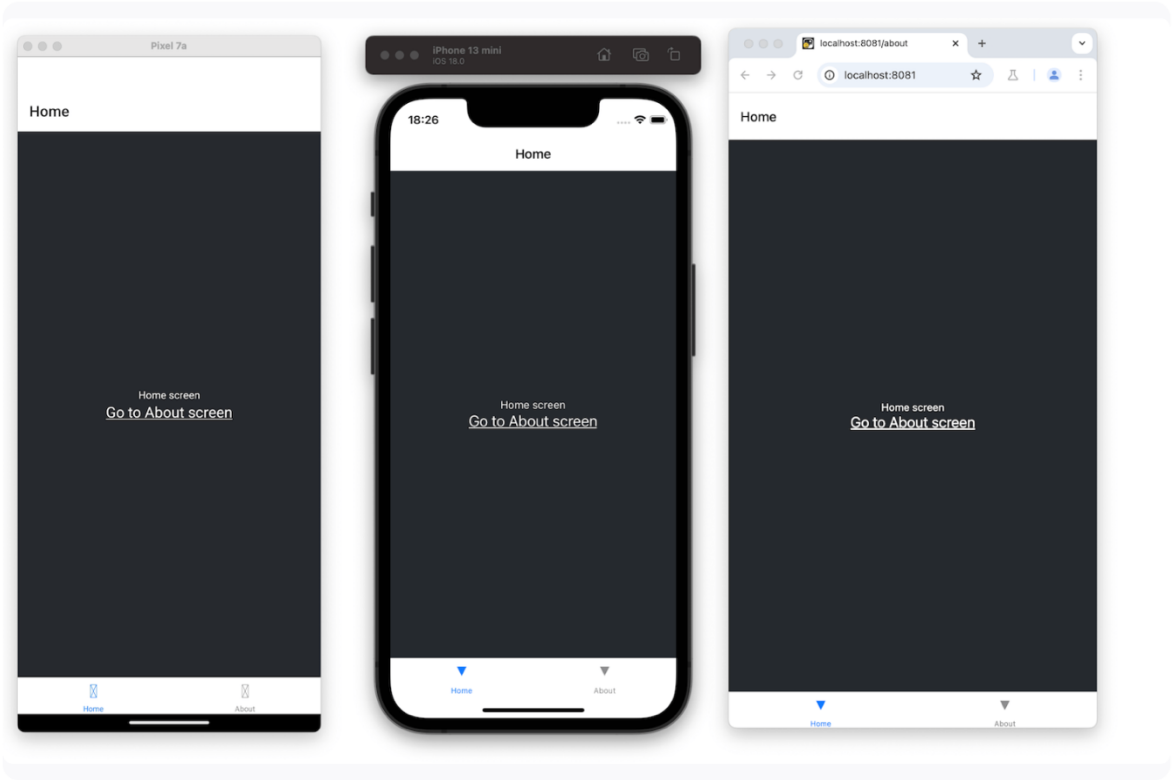
app/_layout.tsx

```tsx
import { Stack } from 'expo-router';

export default function RootLayout() {
  return (
    <Stack>
      <Stack.Screen name="(tabs)" options={{ headerShown: false }} />
    </Stack>
  );
}
```

Всередині `(tabs)/_layout.tsx` додайте компонент `Tabs`, щоб визначити нижню панель табів:

app/(tabs)/_layout.tsx

```
import { Tabs } from 'expo-router';

export default function TabLayout() {
  return (
    <Tabs>
      <Tabs.Screen name="index" options={{ title: 'Home' }} />
      <Tabs.Screen name="about" options={{ title: 'About' }} />
    </Tabs>
  );
}
```



Оновлення зовнішнього вигляду нижньої таб-навігації

Зараз нижня таб-навігація виглядає однаково для всіх платформ, але вона не відповідає стилю нашого застосунку. Наприклад, іконки вкладок не налаштовані, фон таб-бару не співпадає з фоном застосунку.

Змінемо `(tabs)/_layout.tsx` файл, щоб додати іконки для вкладок:

1. Імпортуйте набір іконок `Ionicons` з пакета `@expo/vector-icons`.
2. Додайте `tabBarIcon` до маршрутів `index` та `about`. Ця функція отримує параметри `{ color, focused }` і повертає компонент

іконки. З іконного набору можна вибрати ім’я іконки залежно, чи вкладка активна чи ні.

3. Додайте `screenOptions.tabBarActiveTintColor` у компонент `Tabs` і встановіть значення `'#ffd33d'`. Це змінить колір іконки та підпису вкладки, коли вона активна.

Ось оновлений код:

app/(tabs)/_layout.tsx

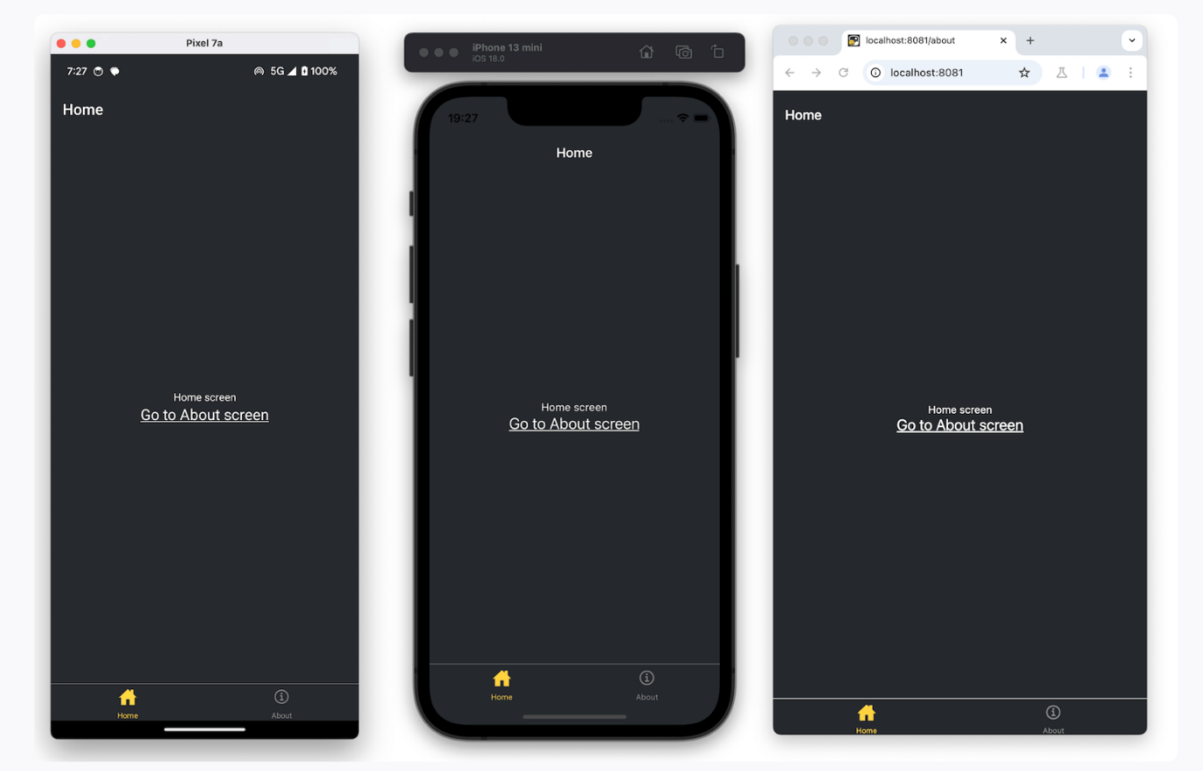
```
import { Tabs } from 'expo-router';
import Ionicons from '@expo/vector-icons/Ionicons';

export default function TabLayout() {
  return (
    <Tabs
      screenOptions={{
        tabBarActiveTintColor: '#ffd33d',
      }}
    >
      <Tabs.Screen
        name="index"
        options={{
          title: 'Home',
          tabBarIcon: ({ color, focused }) => (
            <Ionicons name={focused ? 'home-sharp' : 'home-outline'} color={color} size={24} />
          ),
        }}
      />
      <Tabs.Screen
        name="about"
        options={{
          title: 'About',
          tabBarIcon: ({ color, focused }) => (
            <Ionicons name={focused ? 'information-circle' : 'information-circle-outline'} color={color} size={24}/>
          ),
        }}
      />
    </Tabs>
  );
}
```

Також змінимо фон таб-бару та заголовків за допомогою `screenOptions`:

```
<Tabs
  screenOptions={{
    tabBarActiveTintColor: '#ffd33d',
    headerStyle: {
      backgroundColor: '#25292e',
    },
    headerShadowVisible: false,
    headerTintColor: '#fff',
    tabBarStyle: {
      backgroundColor: '#25292e',
    },
  }}
>
```

У цьому коді:

- `headerStyle.backgroundColor` встановлює фон заголовку як `#25292e`.
- `headerShadowVisible: false` вимикає тінь під заголовком.
- `headerTintColor` задає колір тексту в заголовку (`#fff`).
- `tabBarStyle.backgroundColor` задає фон нижньої панелі табів.



Підсумок

- Ми успішно додали **stack-навігатор** та **панель з вкладками** до нашого застосунку.
- Перенесли екрани Home і About у таб-групу, додали іконки та стилізацію, налаштували загальний лейаут через сторінкові маршрути та каталог `(tabs)`.