

**JS**



# NATIVE DEVICE FEATURES



**JS**





# Agenda

Workflow & Dev Tips

Ionic Post App

Capacitor JS & Native API's

Build for Platforms

Native Device Features & Plugins

# Today's Apps

Ionic Post App & Improvements

Deploy your Ionic Post App

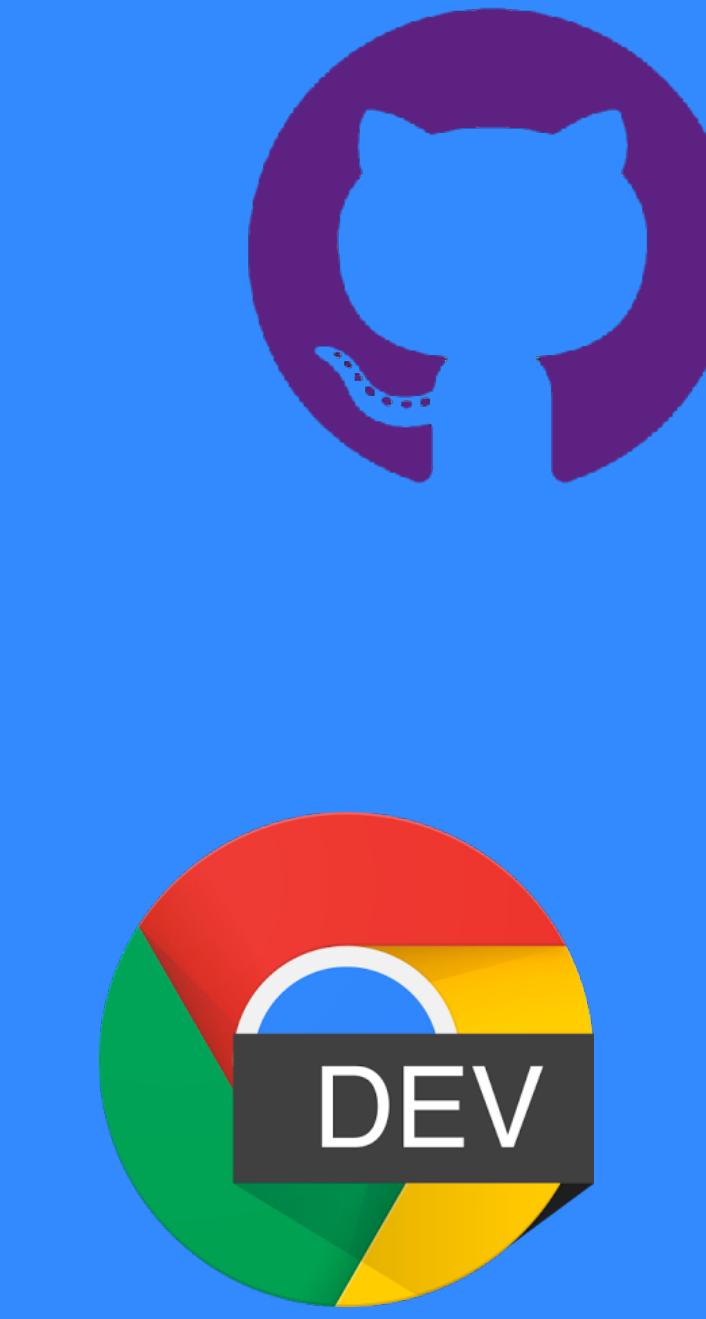
Ionic Camera App

Ionic Post App w/  
@capacitor/camera API

Ionic Post App w/  
@capacitor/storage



# Workflow



# Workflow & Tips

Open in Integrated Terminal || cd into projects

Run existing projects && npm install

Ionic CLI: ionic start && ionic serve

Keybindings

Auto imports / fix imports

Ionic Dev Tips: <https://ionicframework.com/docs/developing/tips>

# Ionic Post App

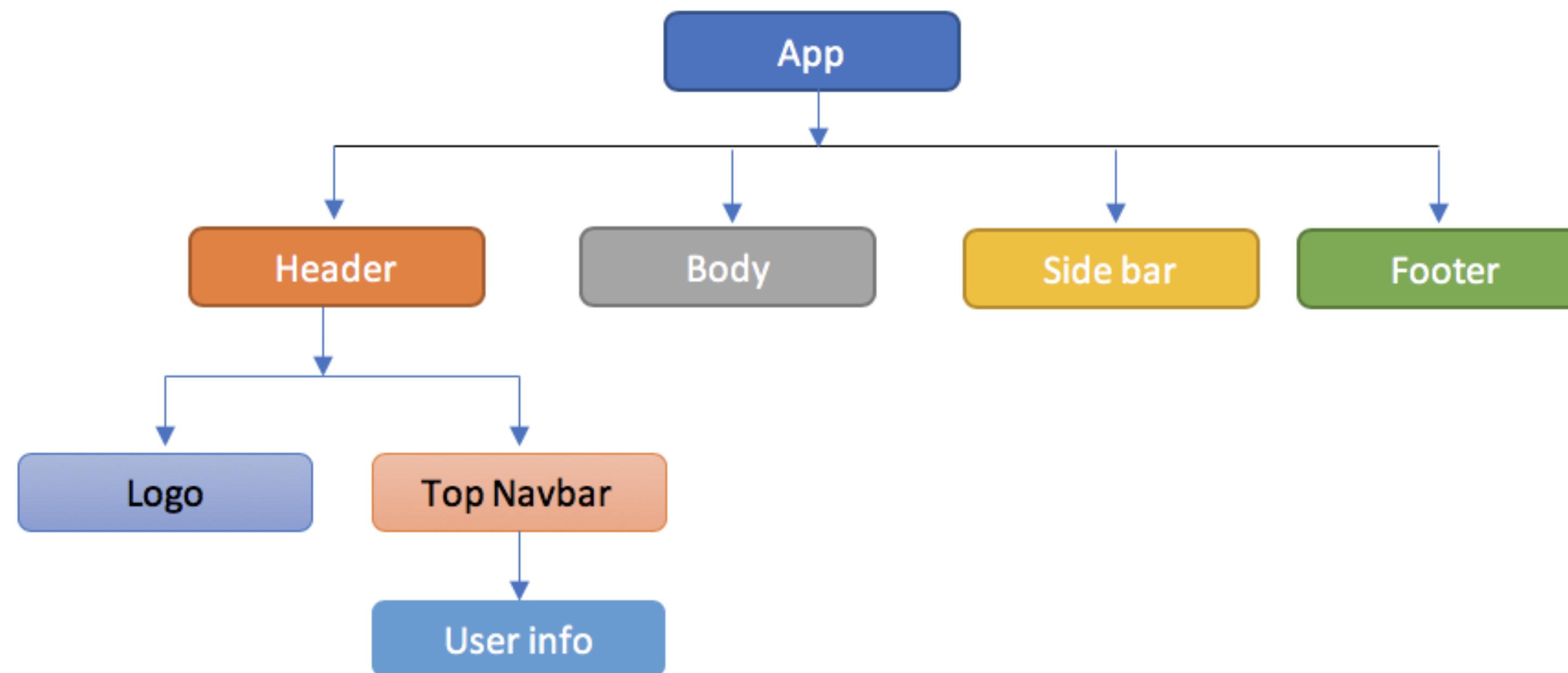
ionic-post-app\*

# Thinking in React

In React, UI is a **function of props & states**. The UI is built with (function) **Components**.

src
└ components
⚛️ PostListItem.jsx
⚛️ UserCard.jsx
⚛️ UserListItem.jsx
└ pages
⚛️ AddPage.jsx
# PostsPage.css
⚛️ PostsPage.jsx
⚛️ UserPage.jsx
⚛️ UsersPage.jsx
└ services
JS postsService.js
JS usersService.js
> theme
⚛️ App.test.tsx
TS App.tsx
TS index.tsx

# APP STRUCTURED IN COMPONENTS



# UI Components

The screenshot shows a web browser window displaying the Ionic UI Components documentation at [ionicframework.com/docs/components](https://ionicframework.com/docs/components). The page has a dark header with the Ionic logo and navigation links for Guide, Components (which is the active tab), CLI, Native, and an Upgrade Guide. The main content area features a large title "UI Components" and a paragraph explaining that Ionic apps are built using high-level building blocks called Components. Below this, there's a grid of cards, each representing a different UI component:

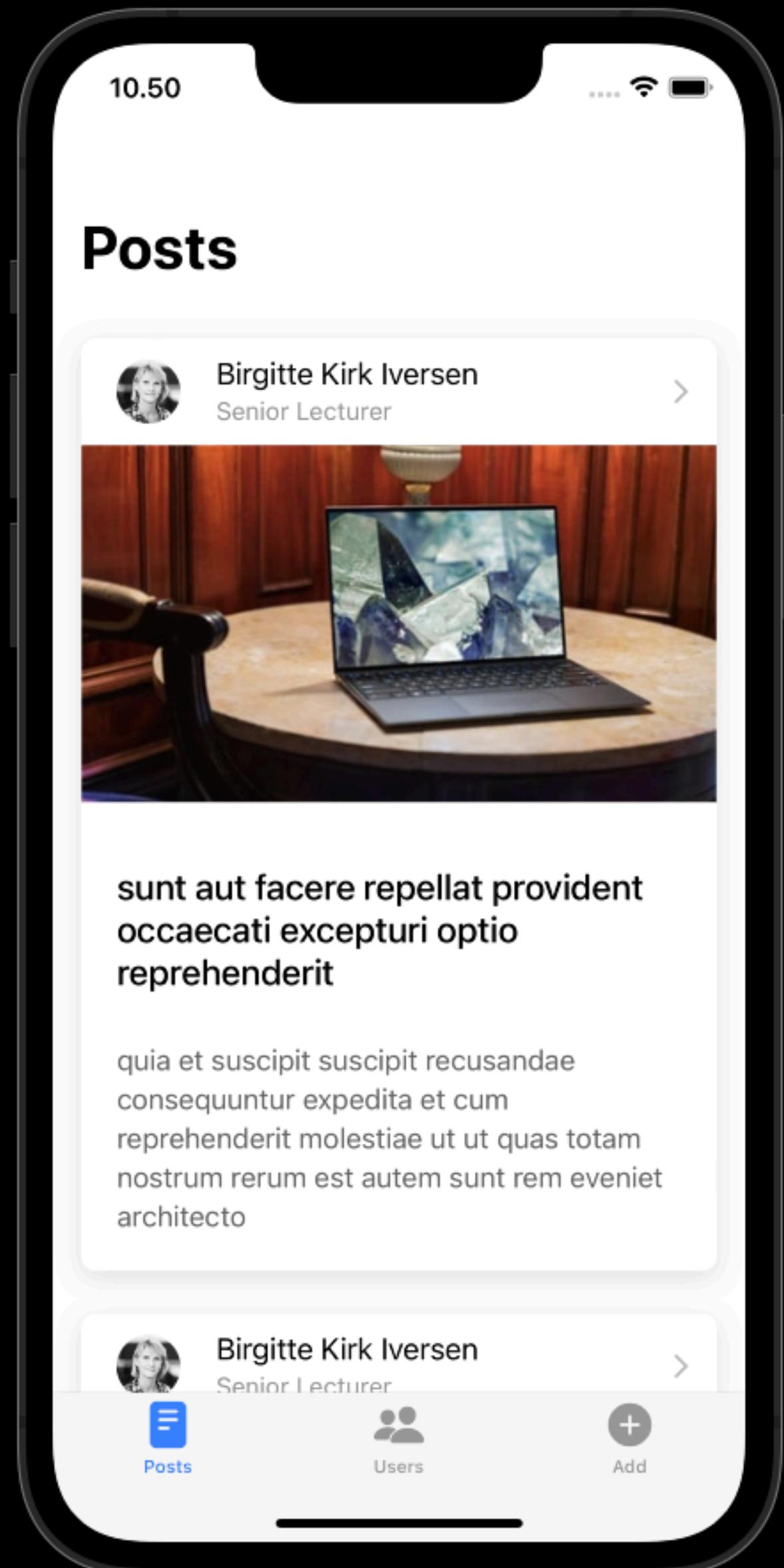
- Action Sheet**: Shows a card with three horizontal bars. Description: Action Sheets display a set of options with the ability to confirm or cancel an action.
- Alert**: Shows a card with a list icon and a red badge with the number 2. Description: Alerts are a great way to offer the user the ability to choose a specific action or list of actions.
- Badge**: Shows a card with a red badge icon. Description: Badges are a small component that typically communicate a numerical value to the user.
- Button**: Shows a card with a blue button icon. Description: Buttons let your users take action. They're an essential way to interact with and navigate through an app.
- Card**: Shows a card with a card icon. Description: Cards are a great way to display an important piece of content, and can contain images, buttons, text, and more.
- Checkbox**: Shows a card with a checked checkbox icon. Description: Checkboxes can be used to let the user know they need to make a binary decision.
- Chip**: Shows a card with a chip icon. Description: Chips are a compact way to display data or actions.
- Content**: Shows a card with a smartphone icon. Description: Content is the quintessential way to interact with and navigate through an app.

# Ionic Post App

9. Improve your post view by adding user details (post create by...). Start by adding hardcoded user details like user avatar and user name using Ionic UI Components.

10. Implement logic to concatenate the posts array with user data from users array. All post objects have a `uid` property.

11. Use the concatenated array to display posts with user details dynamically.



# Combine posts with users

```
import userService from "./usersService";

class PostService {
  >   constructor() { ... }

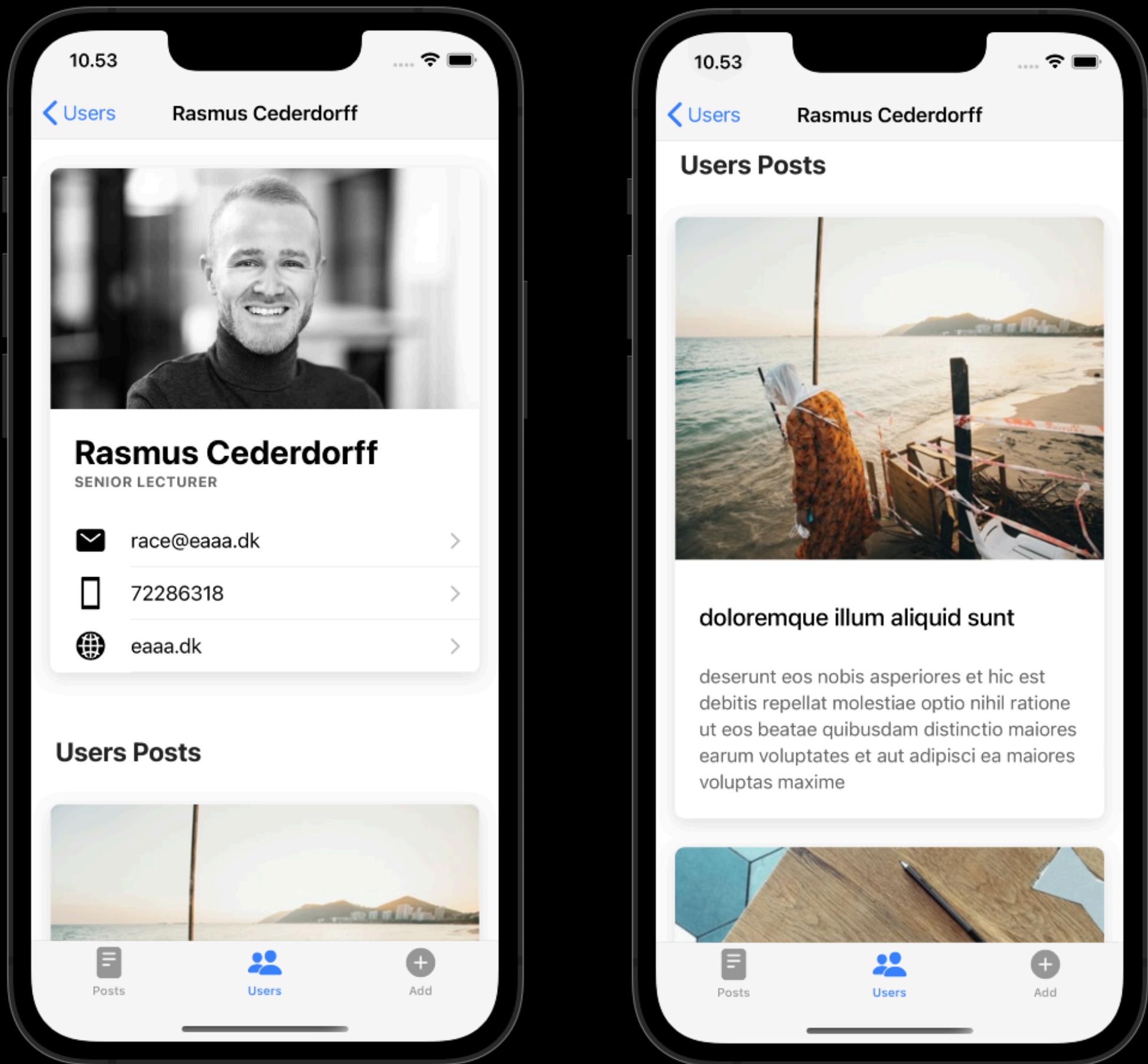
  >   async fetchPosts() { ... }

  >   async getPosts() { ... }

  async getPostsWithUserDetails() {
    if (this.posts.length === 0) {
      await this.fetchPosts();
    }
    const users = await userService.getUsers();

    const postsWithUser = this.posts.map(post => {
      const user = users.find(user => user.id === post.uid);
      post = { ...post, user: user }; // combine objects with spread operator
      delete post.uid; // delete uid - it's inside post.user.id
      return post;
    });
    return postsWithUser;
  }
}
```

# Ionic Post App



12. Improve the user detail view by displaying all posts created by the user. Again, think of the `uid` property on every post object and filter posts by given `uid`.

13. Components, components, components. When implementing improvements, think of how to (re-)use your existing components. No code duplicating is allowed.

# Ionic Post App Improvements

Review your app and make sure to “think in React”. Use reusable components like PostListItem, UserCard, UserListItem, etc.

Create services to handle your logic for posts and users. Export and import services in needed components.



# Ionic Post App

14. Implement the missing tab, Add.

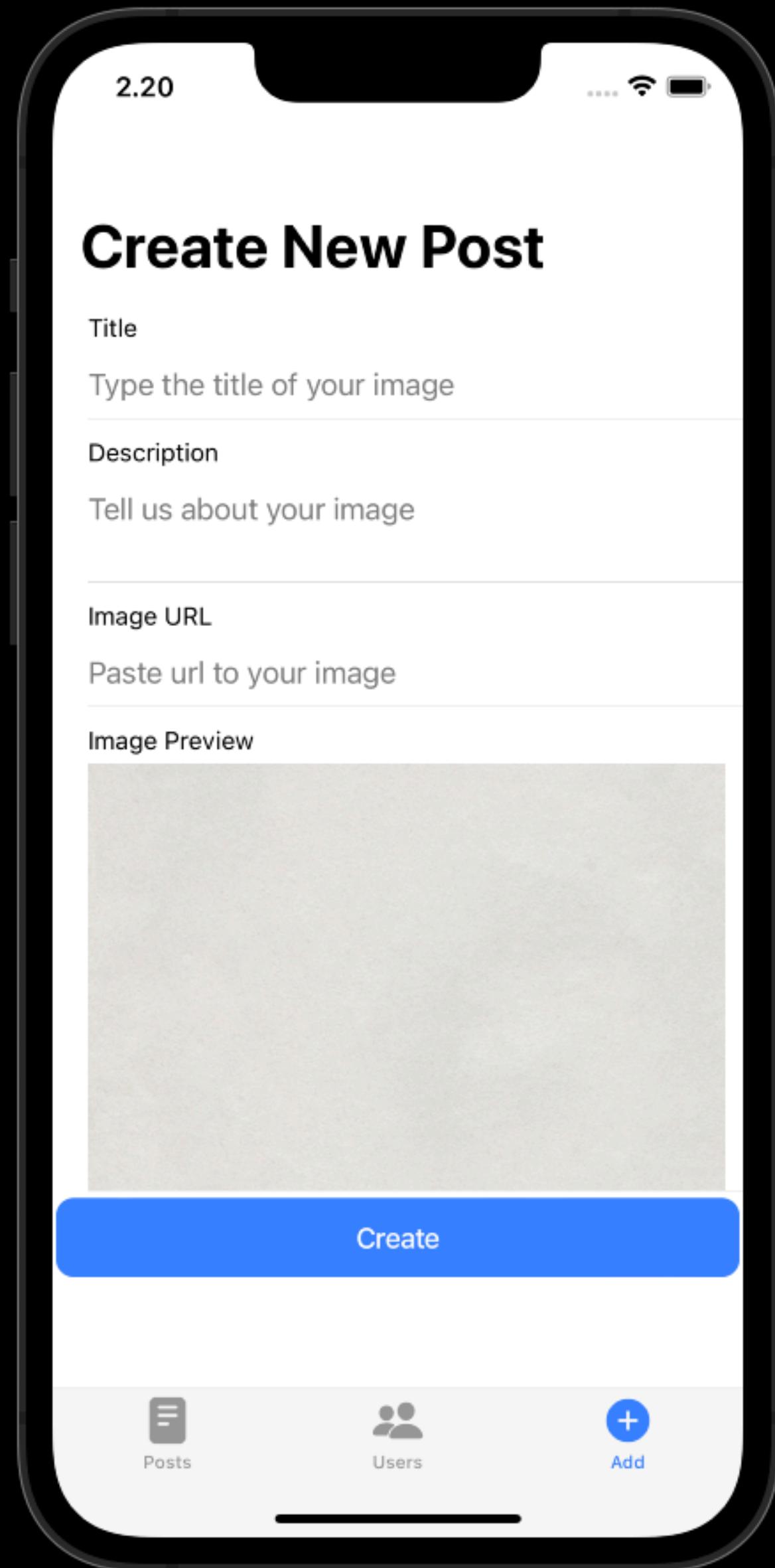
15. Explore Ionic UI Components like `IonInput`, `IonButton` etc.

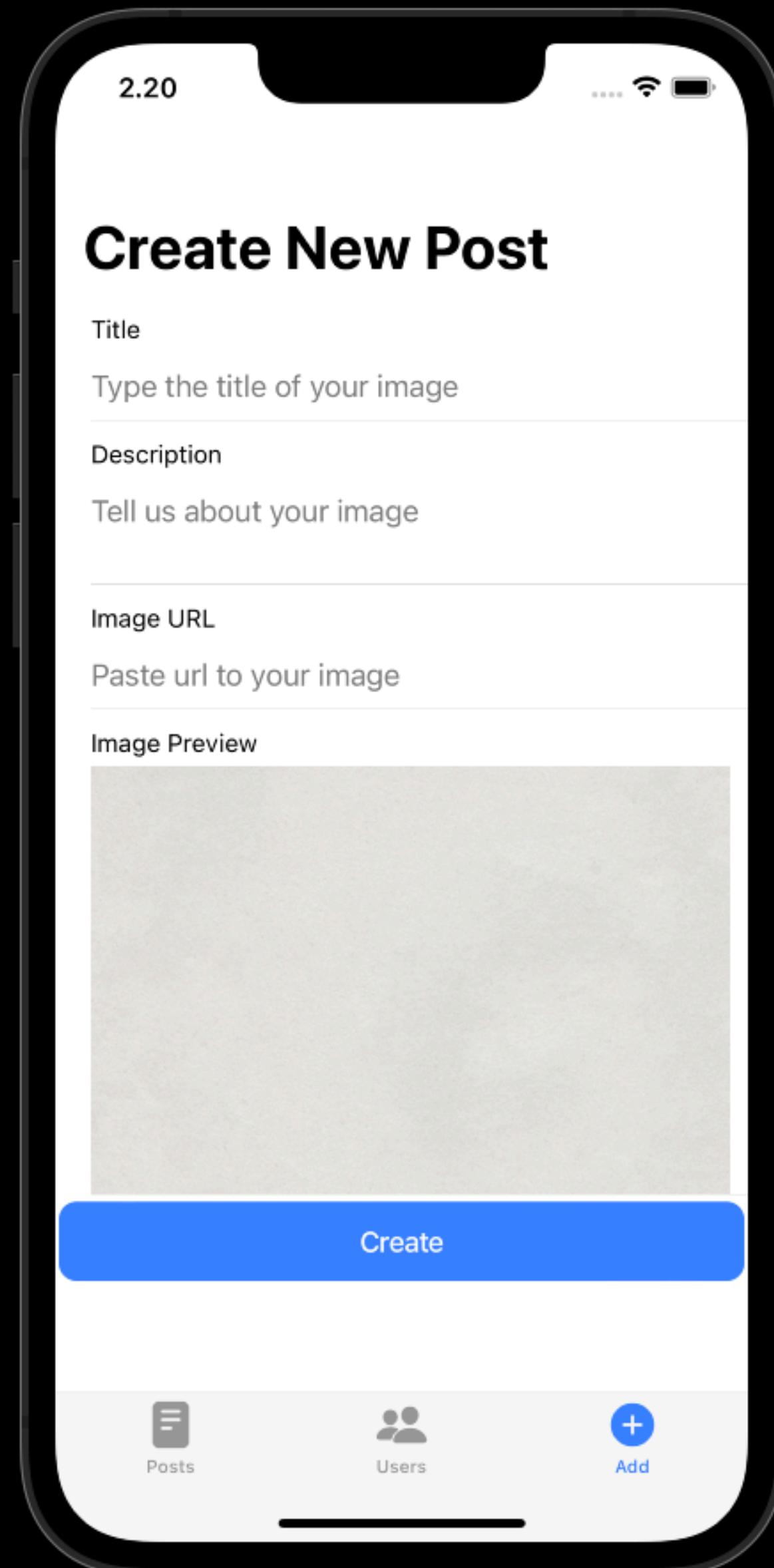
16. Implement the Add tab with UI Components to create a new post object with the following properties: `uid`, `id`, `title`, `body` & `image`.

`uid` can be hardcoded. `image` can be a URL.

Use an `onSubmit` event on the form fired by `IonButton` with type of `submit`. Implement a `handleSubmit` function in App Page Component, creating a new post object with value from inputs (or states).

17. The new post object must be pushed to the array of posts using the posts' service. Make sure to fetch the posts data before pushing the new object.





```
export default function AddPage() {  
  const [title, setTitle] = useState("");  
  const [body, setBody] = useState("");  
  const [url, setUrl] = useState("");  
  
  // ...  
  
  <form onSubmit={handleSubmit}>  
    <IonItem>  
      <IonLabel position="stacked">Title</IonLabel>  
      <IonInput value={title} placeholder="Type the title of your image"  
        onIonChange={e => setTitle(e.target.value)} />  
    </IonItem>  
    <IonItem>  
      <IonLabel position="stacked">Description</IonLabel>  
      <IonTextarea value={body} placeholder="Tell us about your image"  
        onIonChange={e => setBody(e.target.value)}></IonTextarea>  
    </IonItem>  
    <IonItem>  
      <IonLabel position="stacked">Image URL</IonLabel>  
      <IonInput value={url} type="url" placeholder="Paste url to your image"  
        onIonChange={e => setUrl(e.target.value)} />  
    </IonItem>  
    <IonItem>  
      <IonLabel position="stacked">Image Preview</IonLabel>  
      <IonImg src={url === "" ? fallbackUrl : url} />  
    </IonItem>  
    <IonButton type="submit" expand="block">
```

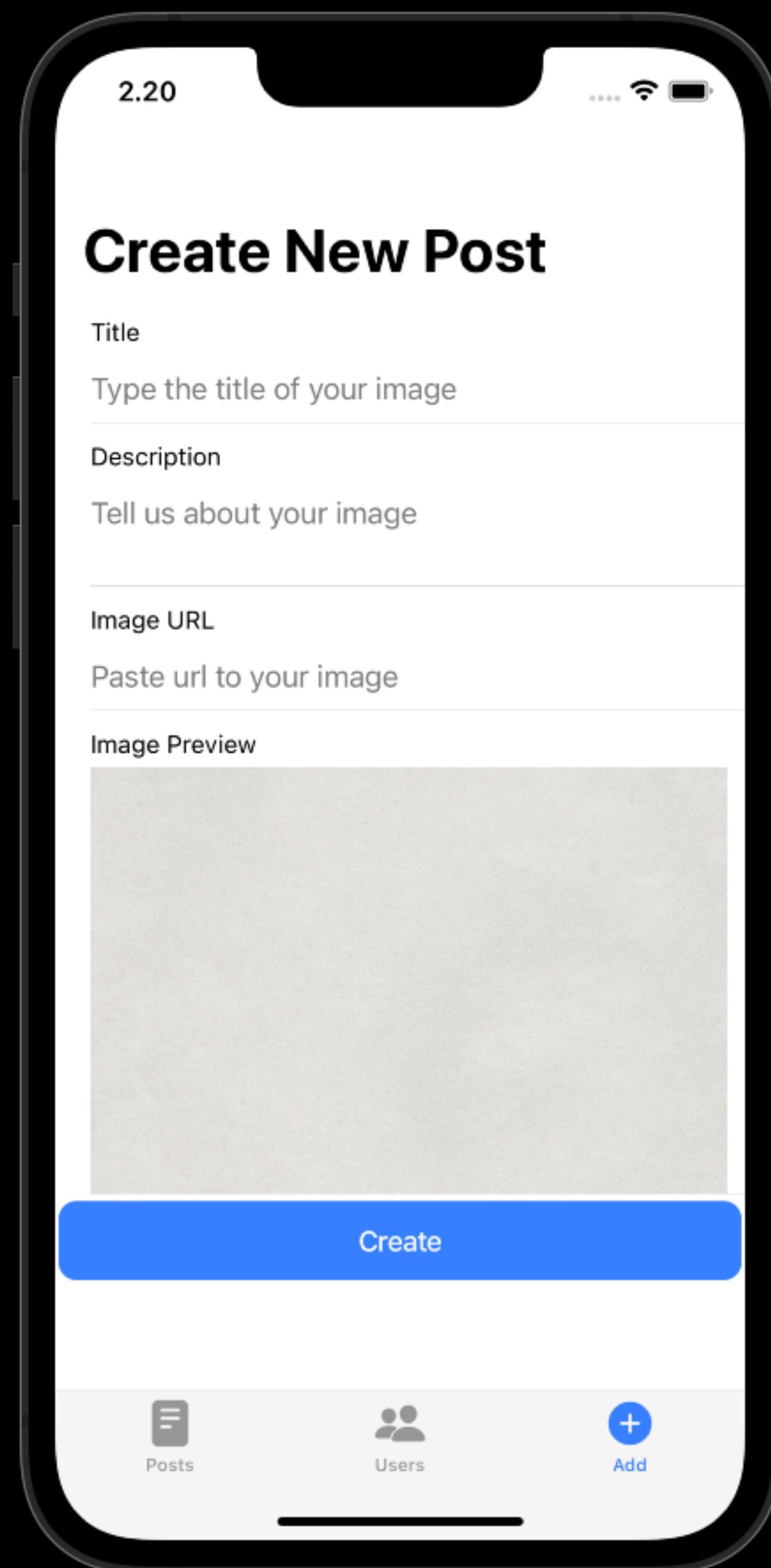
# Form Object

```
export default function FormStateObject() {
  const [formData, setFormData] = React.useState({ firstName: "", lastName: "", mail: "" });

  function handleChange(event) {
    const name = event.target.name;
    const value = event.target.value;
    setFormData(prevFormData => {
      return {
        ...prevFormData,
        [name]: value
      };
    });
  }

  return (
    <form>
      <input type="text" placeholder="First Name" onChange={handleChange} name="firstName" />
      <input type="text" placeholder="Last Name" onChange={handleChange} name="lastName" />
      <input type="email" placeholder="Mail" onChange={handleChange} name="mail" />

      <p>Form data object: {JSON.stringify(formData)}</p>
    </form>
  );
}
```



```
export default function AddPage() {
  const [title, setTitle] = useState("");
  const [body, setBody] = useState("");
  const [url, setUrl] = useState("");
  const fallbackUrl = "https://media.istockphoto.com/photos/white-paper-texture-ba"

  function handleSubmit(event) { ←
    event.preventDefault();

    const newPost = {
      uid: 4,
      title: title,
      body: body,
      image: url
    };
    postService.createPost(newPost);

  }

  return (
    <IonPage>
      <IonHeader>...
      </IonHeader>
      <IonContent fullscreen>
        <IonHeader collapse="condense">...
        </IonHeader>
        <form onSubmit={handleSubmit}>
          <IonItem>
            <IonLabel position="stacked">Title</IonLabel>
```

# Form Object

```
export default function FormStateObject() {
  const [formData, setFormData] = React.useState({ firstName: "", lastName: "", mail: "" });

  function handleChange(event) {
    const name = event.target.name;
    const value = event.target.value;
    setFormData(prevFormData => {
      return {
        ...prevFormData,
        [name]: value
      };
    });
  }

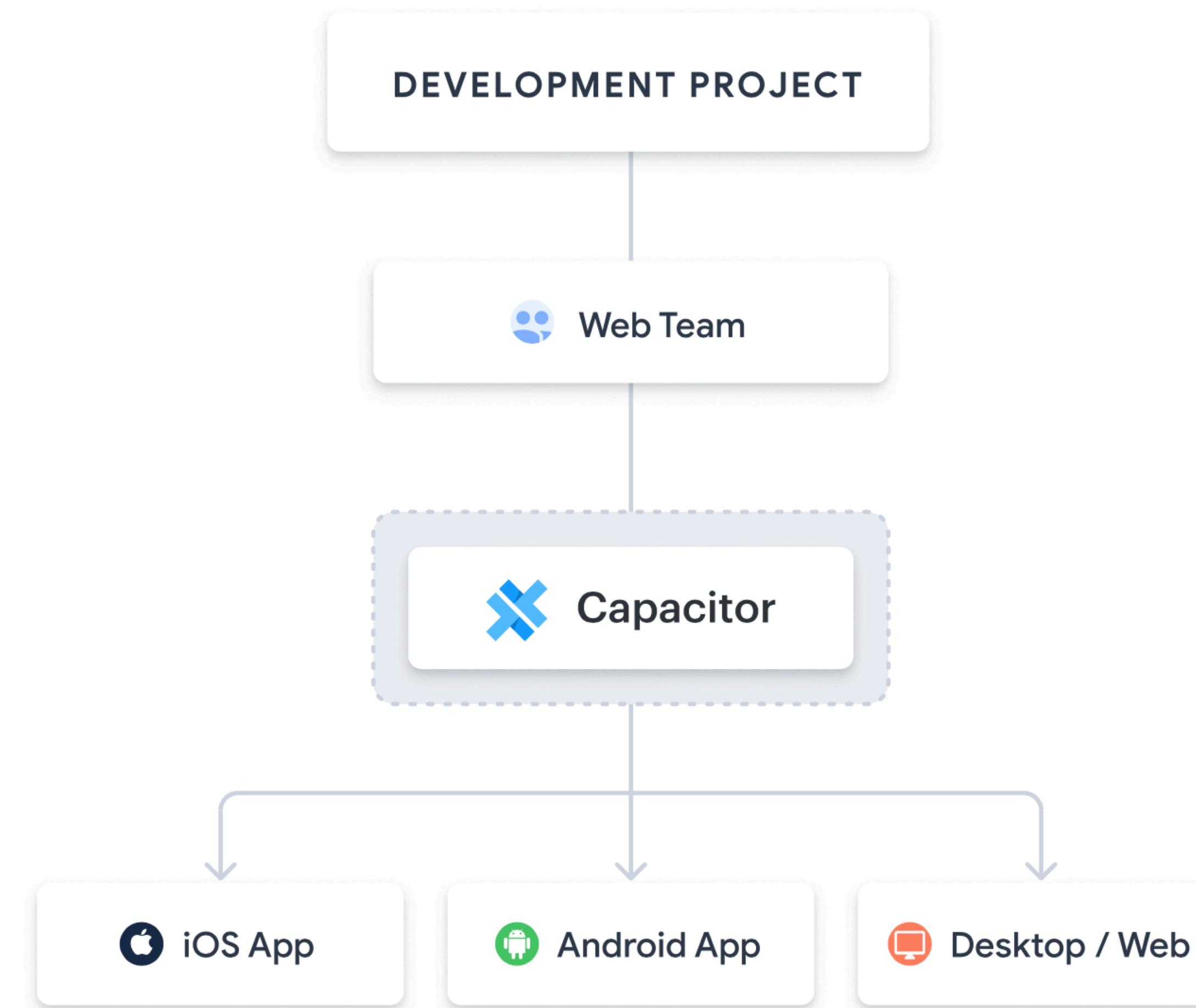
  return (
    <form>
      <input type="text" placeholder="First Name" onChange={handleChange} name="firstName" />
      <input type="text" placeholder="Last Name" onChange={handleChange} name="lastName" />
      <input type="email" placeholder="Mail" onChange={handleChange} name="mail" />

      <p>Form data object: {JSON.stringify(formData)}</p>
    </form>
  );
}
```

# Capacitor JS & Native API's

[Building Cross-platform Apps with Capacitor](#)

# Capacitor JS Project Flow



# Web Native

“Web Native is the idea that teams should build modern Web Apps and combine them with tools like Capacitor that unlock powerful Native APIs to the app for the platform its running on.”

“Web Native is the full capability and access to developers of the Web Platform, with the full functionality and performance benefits of traditional native apps.”

“Web Native is "hybrid" done right, and it's the future of mobile app development.”

<https://webnative.tech/>



# Back in the 2010s (Cordova)

Feature	Native	Web-only	Hybrid
Device Access	Full	Limited	Limited
Performance	High	Medium to High	Low
Development Language	Platform Specific	HTML, CSS, Javascript	HTML, CSS, Javascript
Cross-Platform Support	No	Yes	Yes
User Experience	High	Medium to High	Low
Code Reuse	No	Yes	Yes



Runs web apps  
across multiple  
platforms

Hybrid Apps

# Today (Capacitor)

Feature	Native	Web-only	Hybrid
Device Access	Full	Limited	Full (with plugins)
Performance	High	Medium to High	Medium to High
Development Language	Platform Specific	HTML, CSS, Javascript	HTML, CSS, Javascript
Cross-Platform Support	No	Yes	Yes
User Experience	High	Medium to High	Medium to High
Code Reuse	No	Yes	Yes



Runs modern web apps  
natively on iOS, Android,  
Electron and Web  
multiple platforms.

Web Native Apps & PWA

# Capacitor

A cross-platform native runtime for web apps.



Capacitor is a cross-platform native runtime that makes it easy to build modern web apps that run natively on iOS, Android, and the Web. Representing the next evolution of Hybrid apps, Capacitor creates Web Native apps, providing a modern native container approach for teams who want to build web-first without sacrificing full access to native SDKs when they need it.

# Capacitor

A cross-platform native runtime for web apps.

The Web View and the native app communicate through the use of Capacitor or Cordova plugins. Plugins provide native APIs such as camera, geolocation, and filesystem access to your web app.



UI (Web View)

JS, HTML & CSS

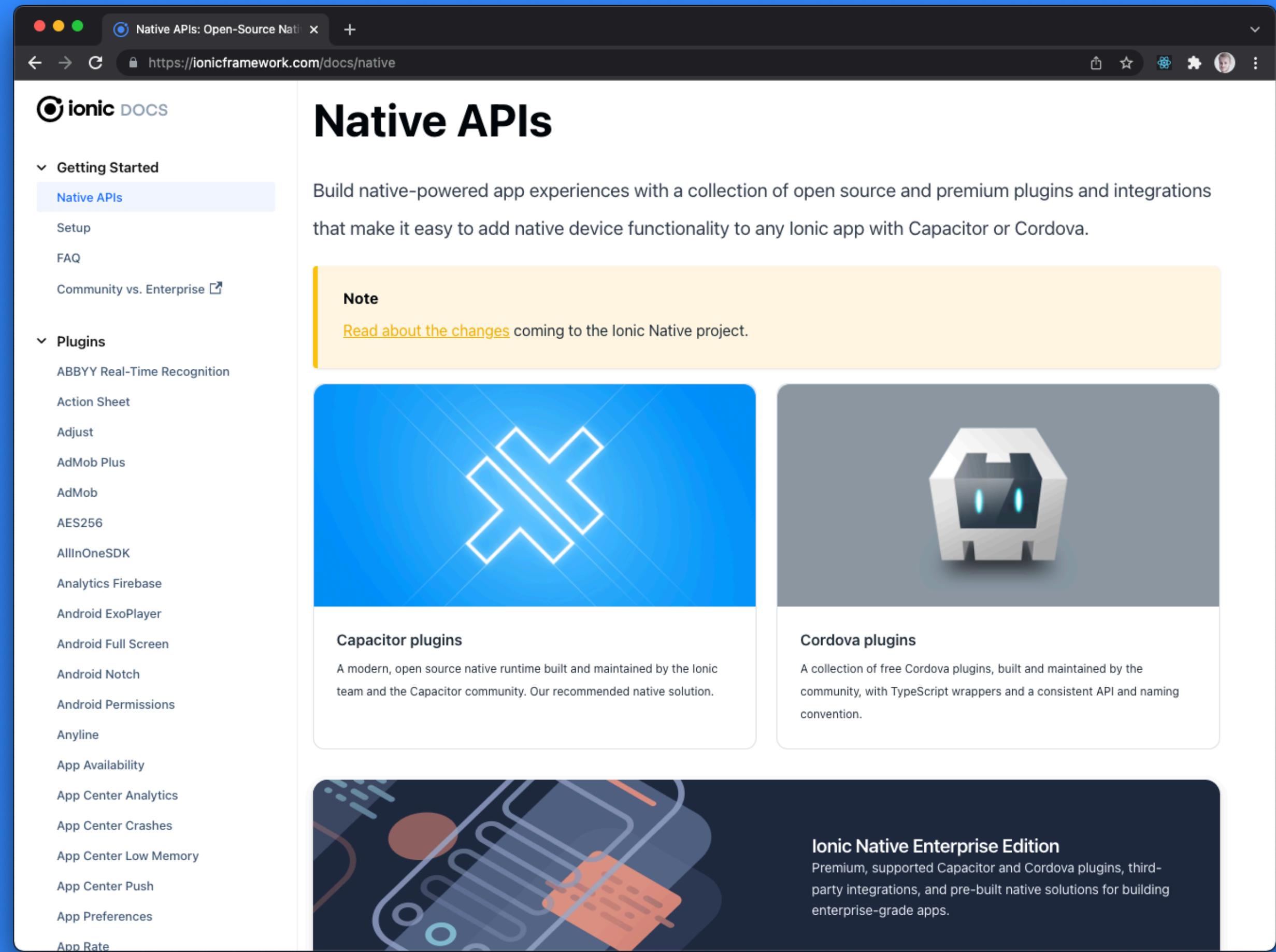
Ionic React

Capacitor

iOS, Android & PWA

# Capacitor Plugins

Enables directly access to native devices features with Native APIs



<https://ionicframework.com/docs/native>

# Build for Platforms

<https://ionicframework.com/docs/react/your-first-app/deploying-mobile>



Well, ~~programming~~ configuration can be hard



# Android Development

[https://ionicframework.com/docs/  
developing/android](https://ionicframework.com/docs/developing/android)

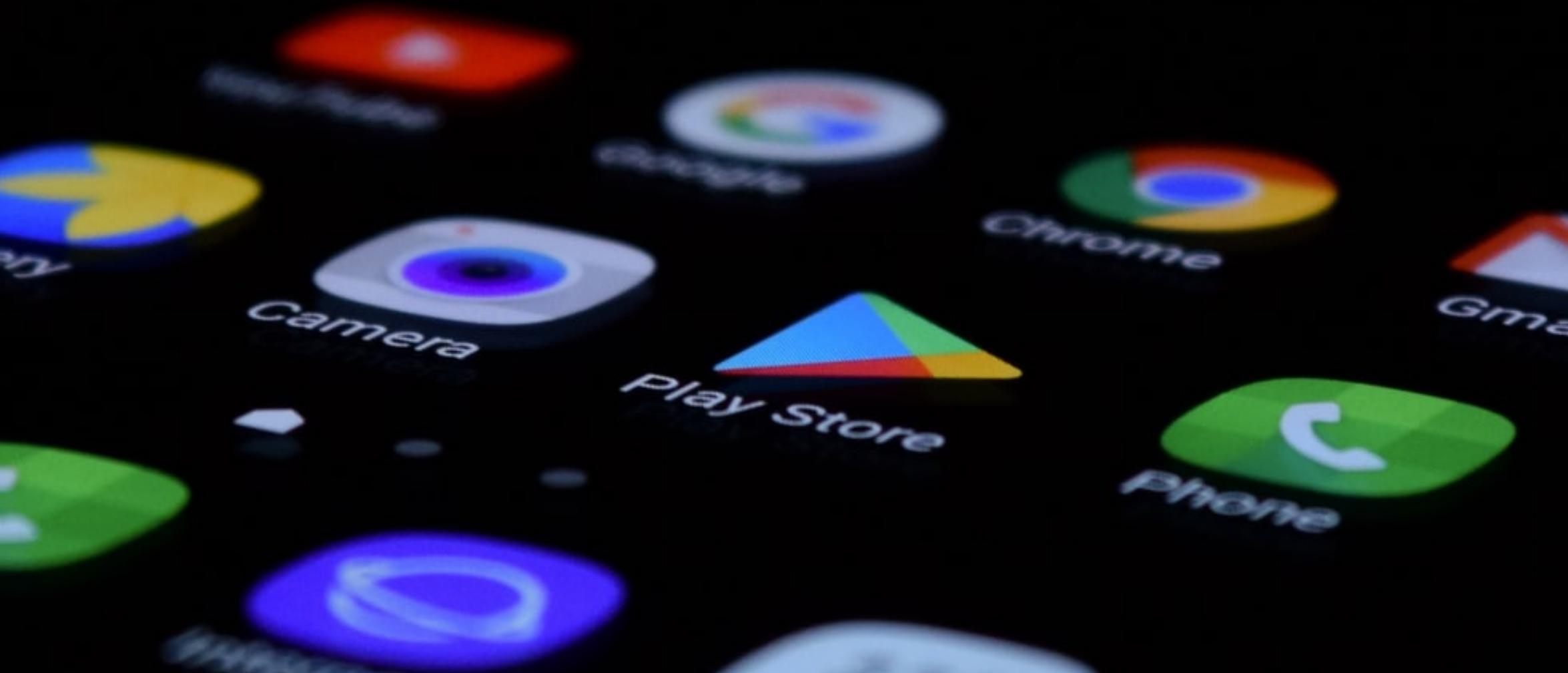


You need Android Studio to  
run on Android emulators &  
devices.

Android can be developed on  
Windows, macOS & Linux.

# Deploy to Android

<https://ionicframework.com/docs/developing/android>



Follow instructions for Capacitor only and use your existing Ionic Post App.

1. Install Android Studio and Android SDK.
2. Configure Command Line Tools (some environment variables must be set).
3. Create a Virtual device and/or set up Android Device.
4. Skip “Cordova Setup” and go to “Project Setup”.
5. Run with Capacitor (and skip “Running with Cordova”).
6. Try out debugging using Chrome DevTools.
7. Move on with Ionic Post App and use live reload to develop and test your app.

# Dev tips

<https://ionicframework.com/docs/developing/tips>

You need a Mac 🤯

Two main workflows:

1. Running with Xcode Xcode
2. Running with Ionic CLI

## iOS Development

<https://ionicframework.com/docs/developing/ios>



Follow instructions for Capacitor only and use your existing Ionic Post App.

1. Download Xcode from AppStore. Make sure you have the latest version.

2. Make sure the command-line tools are selected for use:

```
$ xcode-select --install
```

3. Add iOS with

```
$ ionic capacitor add ios
```

4. Open in Xcode and run with Xcode

5. Run with CLI & Live-reload with Capacitor

6. Debug iOS with Safari

7. Move on with Ionic Post App and use live reload to develop and test your app.

## Deploy to iOS

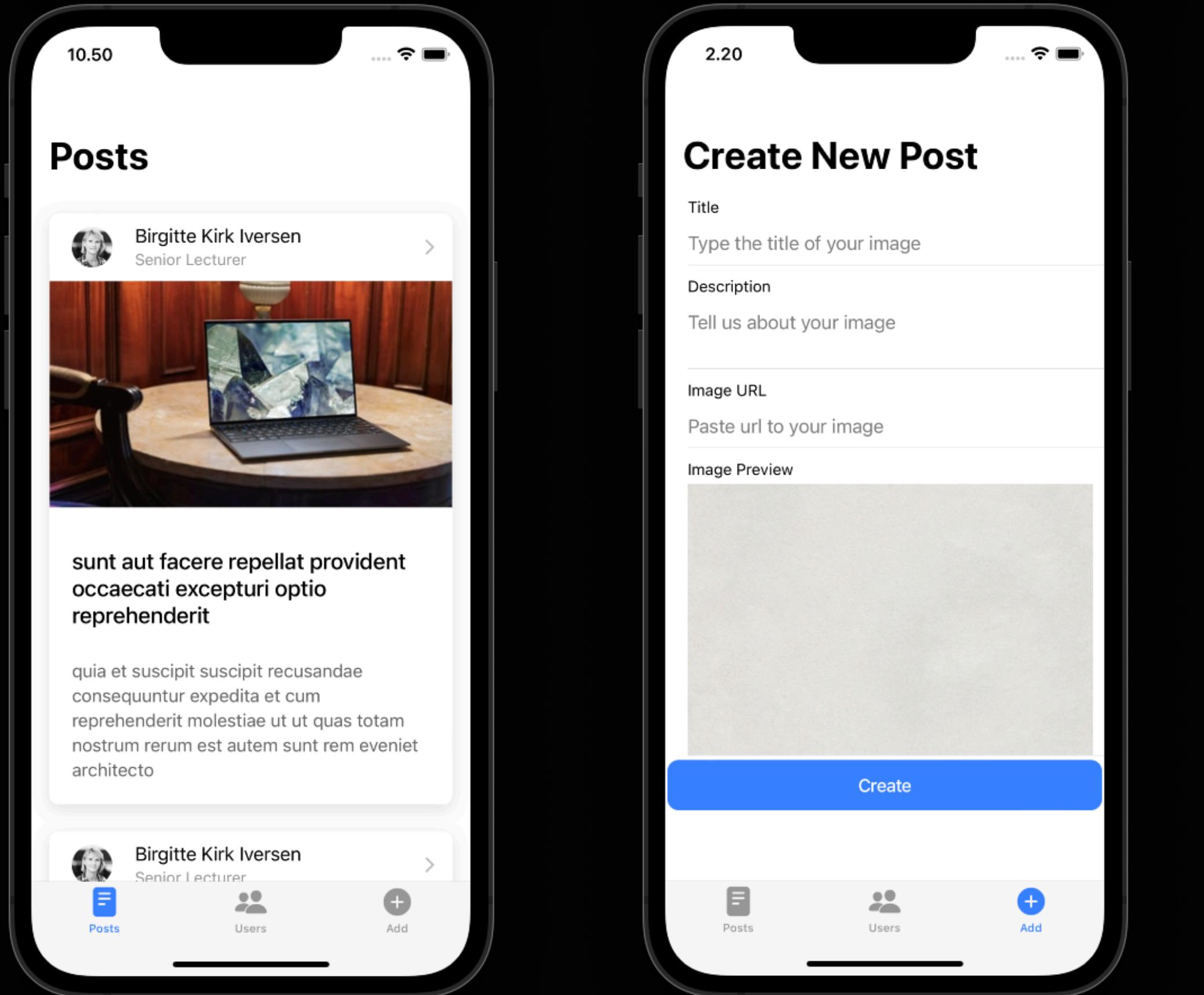
<https://ionicframework.com/docs/developing/ios>



# Live Reload

<https://ionicframework.com/docs/react/your-first-app/live-reload>

# Deploy your Ionic Post App

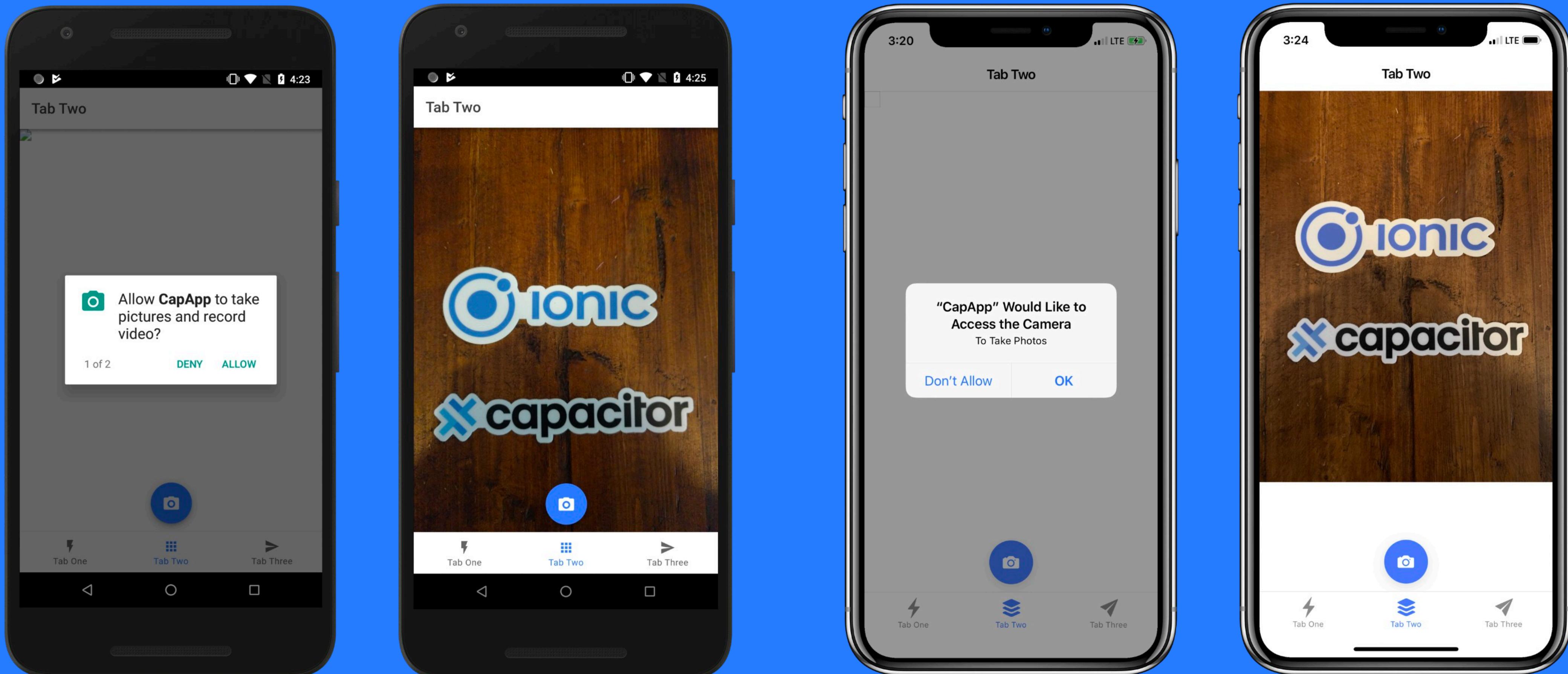


Guides: [Android](#) & [iOS](#)

1. \$ ionic build
2. \$ ionic cap add ios or  
ionic cap add android
3. Run on platforms with  
Android Studio, Xcode or CLI.

# User Permissions

<https://ionicframework.com/docs/react/your-first-app/deploying-mobile>



# Configuring Android

<https://capacitorjs.com/docs/android/configuration>

The screenshot shows a web browser window with the title 'Configuring Android - Capacitor'. The URL in the address bar is <https://capacitorjs.com/docs/android/configuration>. The page content is titled 'Setting Permissions'. It explains that permissions are defined in `AndroidManifest.xml` inside the `<manifest>` tag. An example code snippet shows how to add Network permissions:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.getcapacitor.myapp">
  <activity>
    <!-- other stuff -->
  </activity>

  <!-- More stuff -->

  <!-- Your permissions -->

  <!-- Network API -->
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
</manifest>
```

Below the code, a note states: 'Generally, the plugin you choose to use will ask you to set a permission. Add it in this file.'

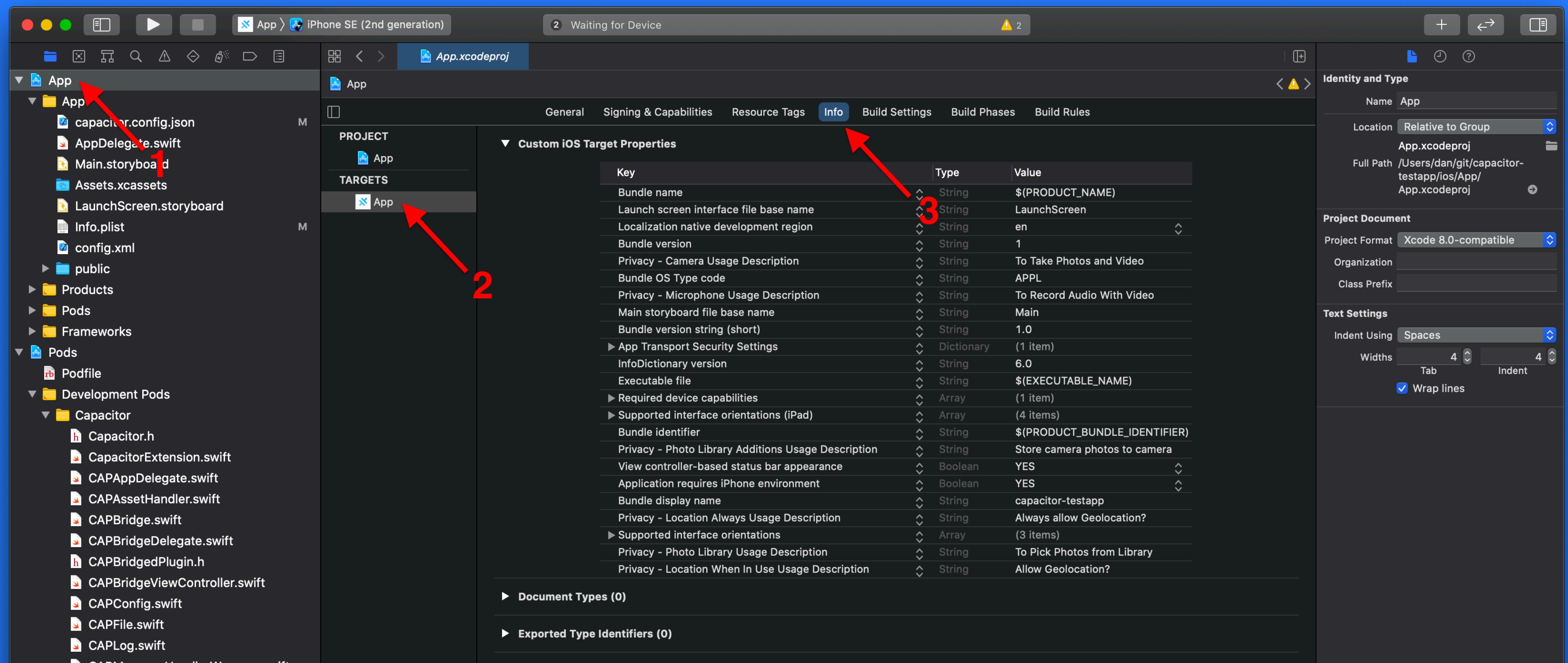
The left sidebar contains navigation links for 'Environment Setup', 'Installation', 'Using with Ionic Framework', 'Basics' (with sub-links for Development Workflow, Using Plugins, Native Project Configuration, and Utilities), 'Upgrade Guides', 'Cordova/PhoneGap', 'Concepts', and 'iOS' (with sub-links for Getting Started, Configuration, Custom Native Code, Deploying to App Store, Custom ViewController, and Troubleshooting).

The top right features a search bar, navigation icons, and links for 'Docs', 'Plugins', 'CLI', 'Community', 'Blog', 'Enterprise', and social media sharing.

The right side of the page includes a 'CONTENTS' sidebar with links to 'Configuring AndroidManifest.xml', 'Changing the Package ID', 'Changing the App Name', 'Deeplinks (aka Android App Links)', 'URL Schemes', and 'Setting Permissions'. There is also a 'Submit an edit' button and an 'appflow' logo with the tagline 'Continuous app delivery made easy. Build, publish, and update from the cloud.'

# Configuring iOS

<https://capacitorjs.com/docs/ios/configuration>

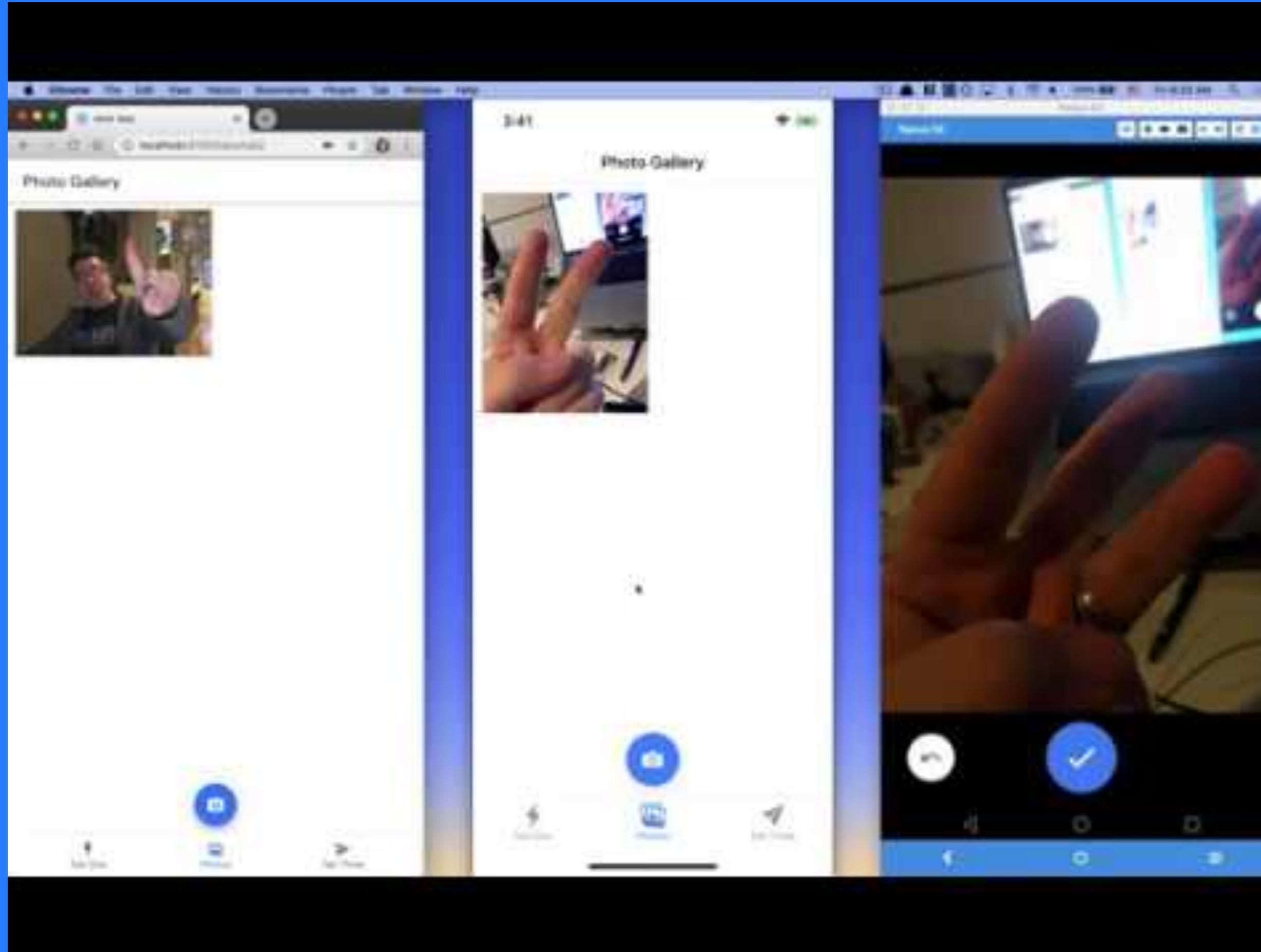


# Native Device Features & Plugins

<https://ionicframework.com/docs/native>

# Ionic Camera App

<https://ionicframework.com/docs/react/your-first-app#create-an-app>



# Ionic Post App w/ @capacitor/camera API

```
import { Camera, CameraResultType } from '@capacitor/camera';

const takePicture = async () => {
  const image = await Camera.getPhoto({
    quality: 90,
    allowEditing: true,
    resultType: CameraResultType.Uri
  });

  // image.webPath will contain a path that can be set as an image src.
  // You can access the original file using image.path, which can be
  // passed to the Filesystem API to read the raw data of the image,
  // if desired (or pass resultType: CameraResultType.Base64 to getPhoto)
  var imageUrl = image.webPath;

  // Can be set to the src of an image now
  imageElement.src = imageUrl;
};
```

1. On the Add Page, replace the URL input with the ability to take a photo with the camera or choose an existing one from the photo album.
2. Explore the @capacitor/camera and get inspired by the Ionic Camera App as well.
3. Install the @capacitor/camera.
4. Use CameraResultType.Base64 and save the output in the image property.

# Ionic Post App w/ @capacitor/storage

```
import { Storage } from '@capacitor/storage';

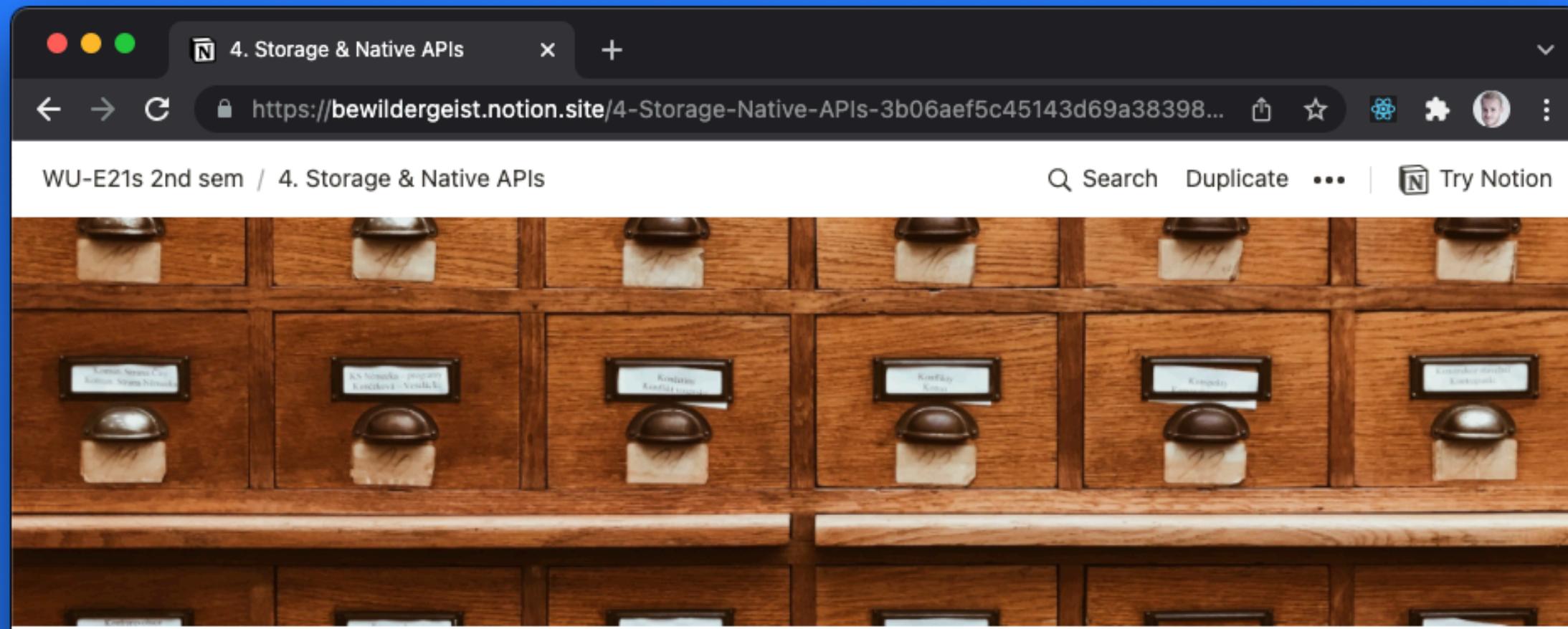
const setName = async () => {
  await Storage.set({
    key: 'name',
    value: 'Max',
  });
}

const checkName = async () => {
  const { value } = await Storage.get({ key: 'name' });
  alert(`Hello ${value}!`);
};

const removeName = async () => {
  await Storage.remove({ key: 'name' });
};
```

1. Explore and install the @capacitor/storage plugin.
2. Reimplement the services with @capacitor/storage. Use the docs to explore how to get and set data.

# Next Thursday



The screenshot shows a Notion page with the following details:

- Title:** 4. Storage & Native APIs
- Page URL:** https://bewildergeist.notion.site/4-Storage-Native-APIs-3b06aef5c45143d69a38398...
- Background Image:** A photograph of a wooden card catalog with multiple drawers, each containing a small card.
- Section Header:** **4. Storage & Native APIs**
- Metadata:**
  - Date: 24/02/2022
  - Teacher: RACE
  - Course: MAD
- Section Header:** **Agenda/ Themes**
- List:**
  - Native Device Features
  - Native APIs
  - Device Storage
  - Cloud Storage
  - Firebase