zalando

Adopting cross-platform React Native at scale

Rene Eichhorn
React Universe 2025

zalando

Founded in 2008 in Berlin, Zalando is building the leading pan-European ecosystem for fashion and lifestyle e-commerce.

~15K

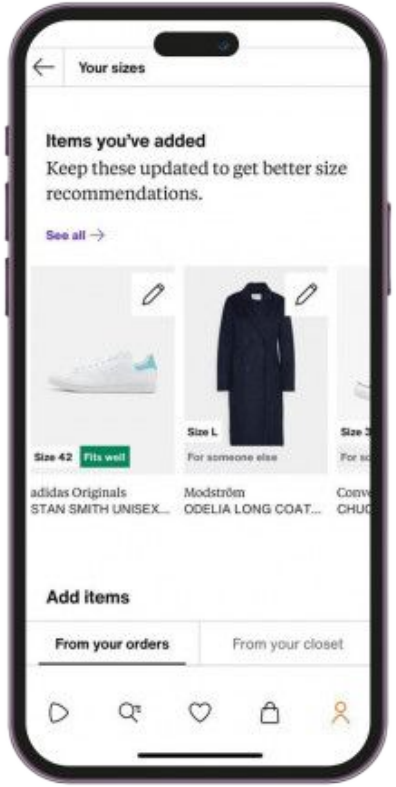
Employees

>50M

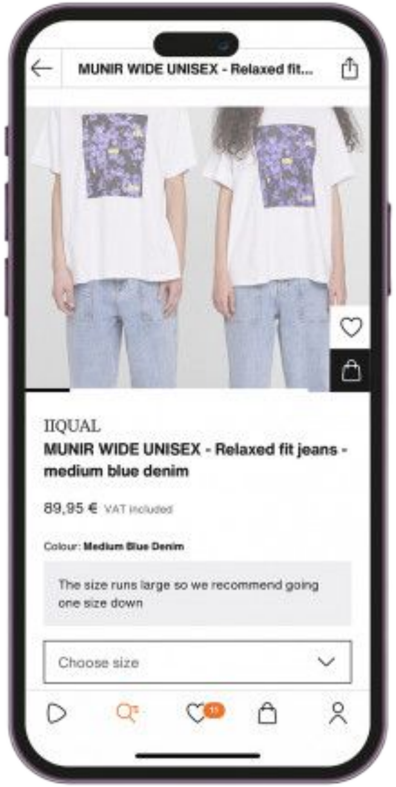
Customers

Zalando App

- Two large codebases for Android and iOS
- roughly **half a million lines of code**
- ~50k pull requests
- 10 year old codebase
- 90+ Screens


Items you've added

Keep these updated to get better size recommendations.


IIQUAL

MUNIR WIDE UNISEX - Relaxed fit jeans - medium blue denim

89,95 € VAT included

Colour: Medium Blue Denim

The size runs large so we recommend going one size down

> the beginning: **Requirements**

01

Faster iterations

Improve developer experience.

Improve development cycles.

Allow quicker and smoother experimentation.

02

Progressive technology adoption

Technology needs to be added not replaced.

Rebuilding the entire app at once is not an option.

Evaluation before committing.

03

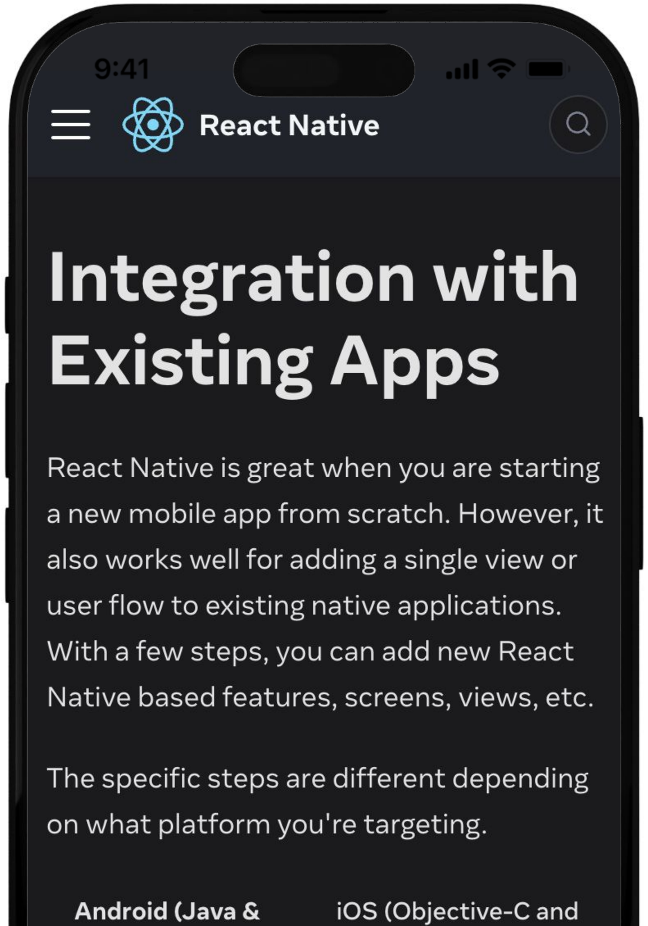
Cross platform *Including Web!*

Shared code for Android, iOS & Web where it makes sense.

More consistent design & user experience, without loosing platform specifics.

Shared concepts with Web!

Progressive Adoption
“Can’t be that difficult right?”



Integration with Existing Apps

React Native is great when you are starting a new mobile app from scratch. However, it also works well for adding a single view or user flow to existing native applications. With a few steps, you can add new React Native based features, screens, views, etc.

The specific steps are different depending on what platform you're targeting.

Android (Java & iOS (Objective-C and

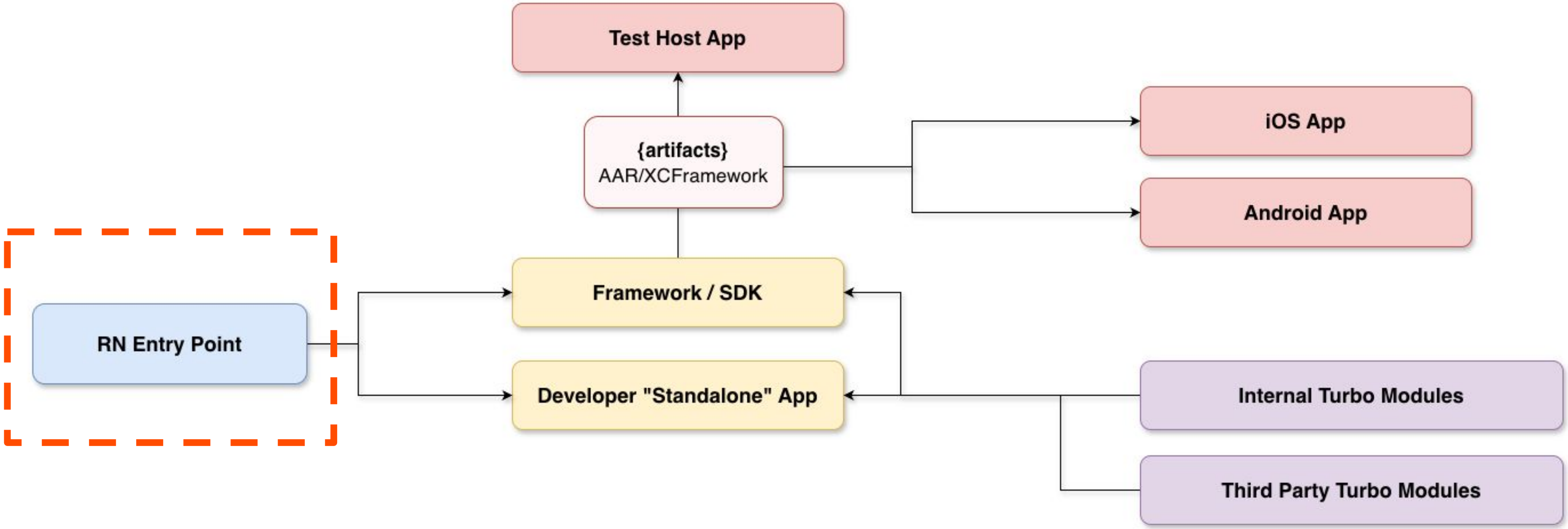
Progressive Adoption

01 **Integration instead of a new application**
Adoption in gradual rollouts and A/B tests

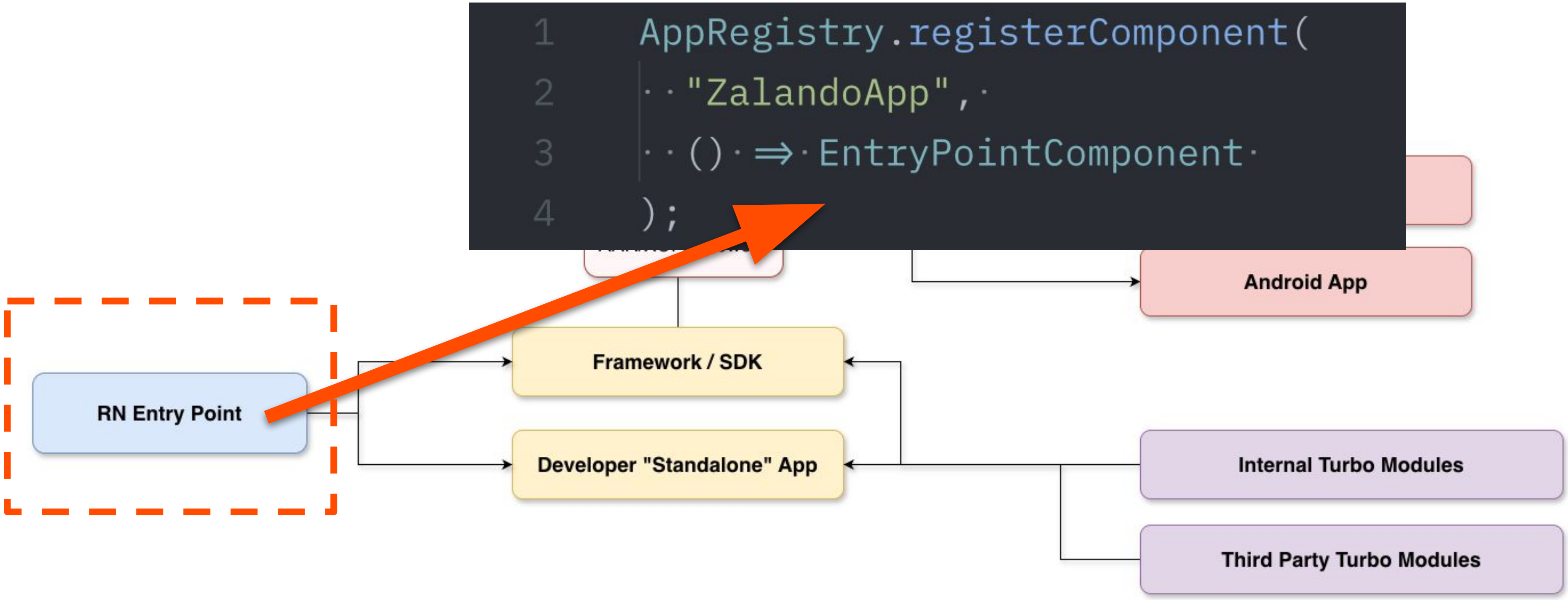
02 **Non-invasive integration**
Ongoing native development must continue as normal

03 **Separation of concerns**
The new architecture must not leak into legacy and vice versa

The Framework

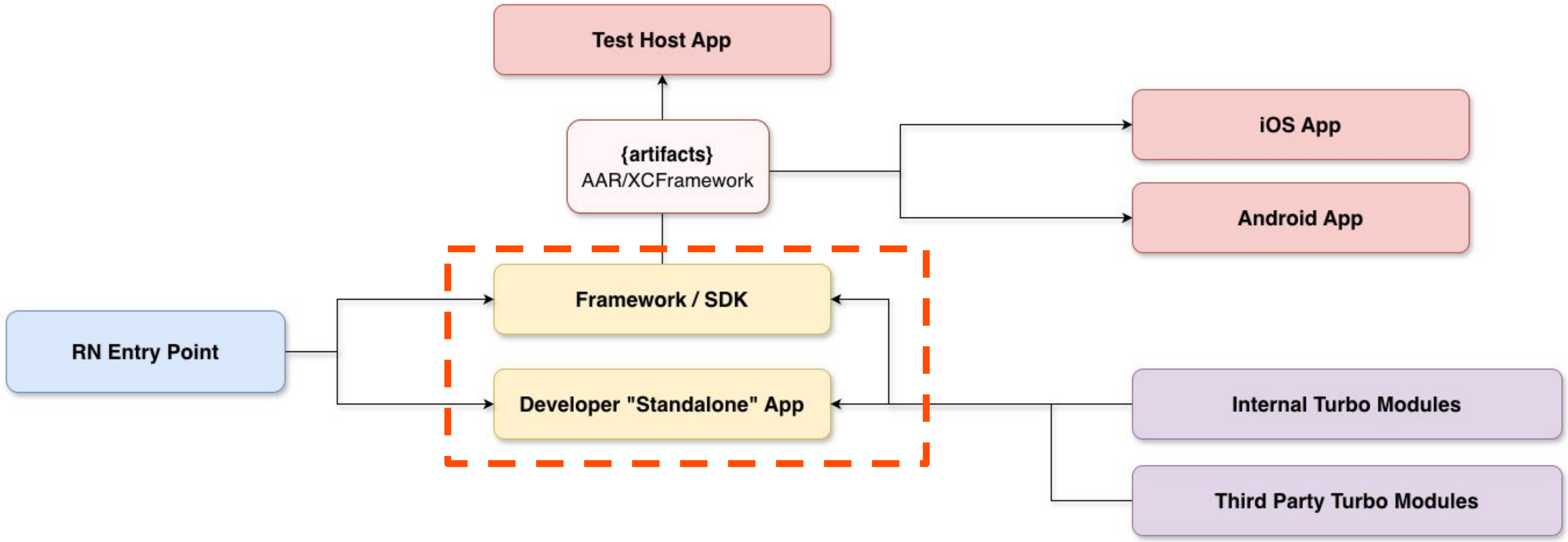


The Framework



```
AppRegistry.registerComponent(
  "ZalandoApp",
  () => EntryPointComponent
);
```

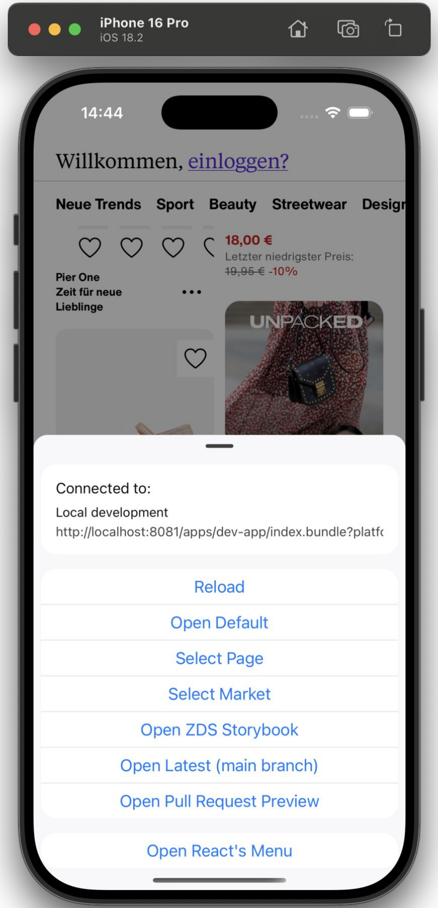
The Framework



01 **Easy way for anyone to get started**
Whether web or native engineer you want an easy as possible way to get development going

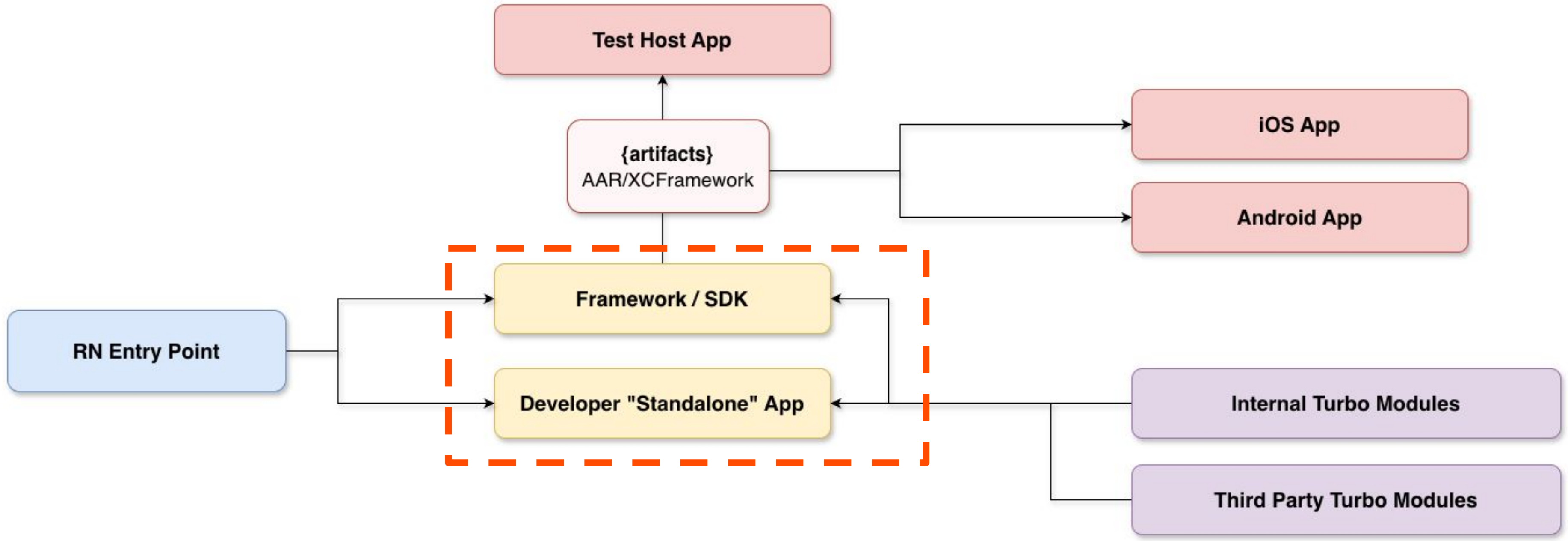
02 **Available as a downloadable app**
Remove the need to compile yourself and getting started quickly.

03 **Testing different environments**
Download JavaScript bundles for staging, production and pull requests.



Connected to:
Local development
http://localhost:8081/apps/dev-app/index.bundle?platf

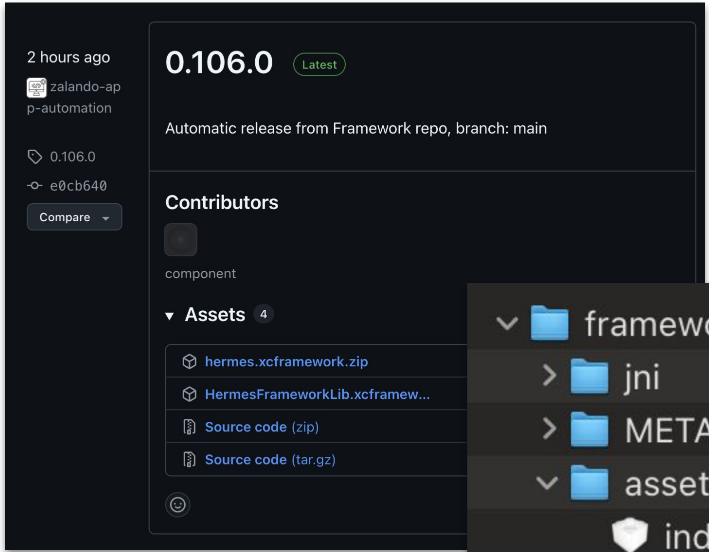
The Framework



Framework Build

- **iOS**: Compile as a XCFramework with
- **Android**: Build as .AAR
- **Public API**:
  - Initialization
  - Creation of views (fragment, activity, view)

Open Sourced by Callstack
[>> github.com/callstack/react-native-brownfield](https://github.com/callstack/react-native-brownfield)

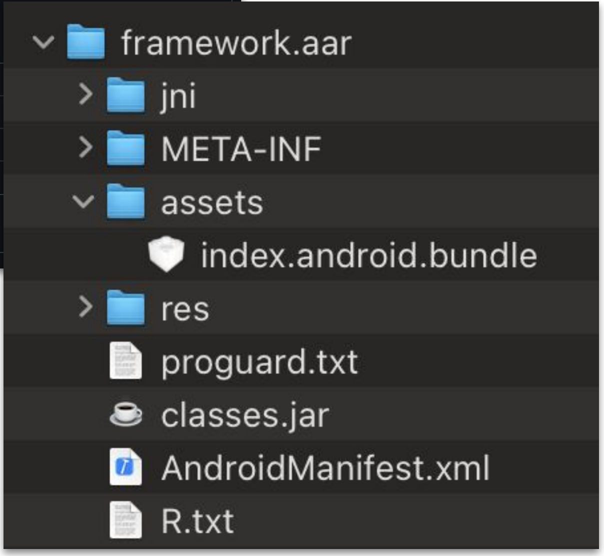

2 hours ago
zalando-app-automation
0.106.0
e0cb640
Compare

0.106.0 Latest

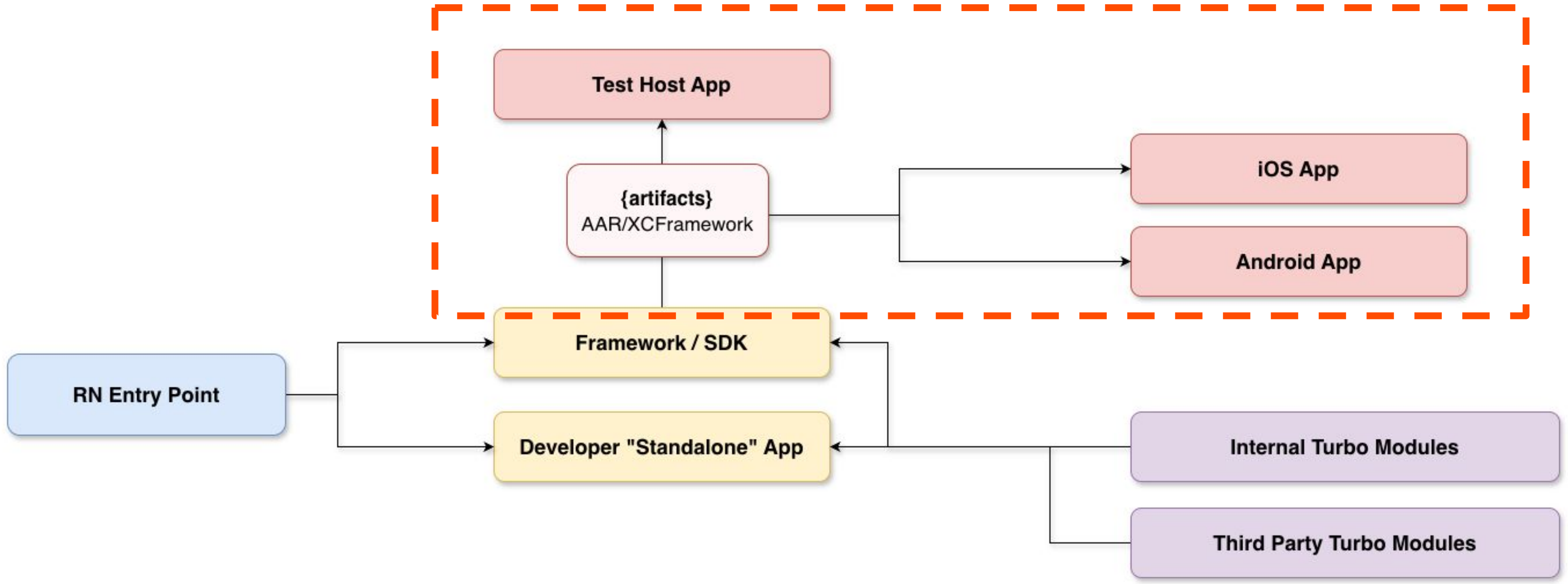
Automatic release from Framework repo, branch: main

Contributors
component

Assets 4
hermes.xcframework.zip
HermesFrameworkLib.xcframew...
Source code (zip)
Source code (tar.gz)


```
framework.aar
  jni
  META-INF
  assets
    index.android.bundle
  res
  proguard.txt
  classes.jar
  AndroidManifest.xml
  R.txt
```

The Framework



Cross platform

How to include web

The cross platform playbook

01 **Platform difference**
Platform differences can be adopted.

02 **Shared Codebase**
“Most” code that doesn’t require platform context is shared

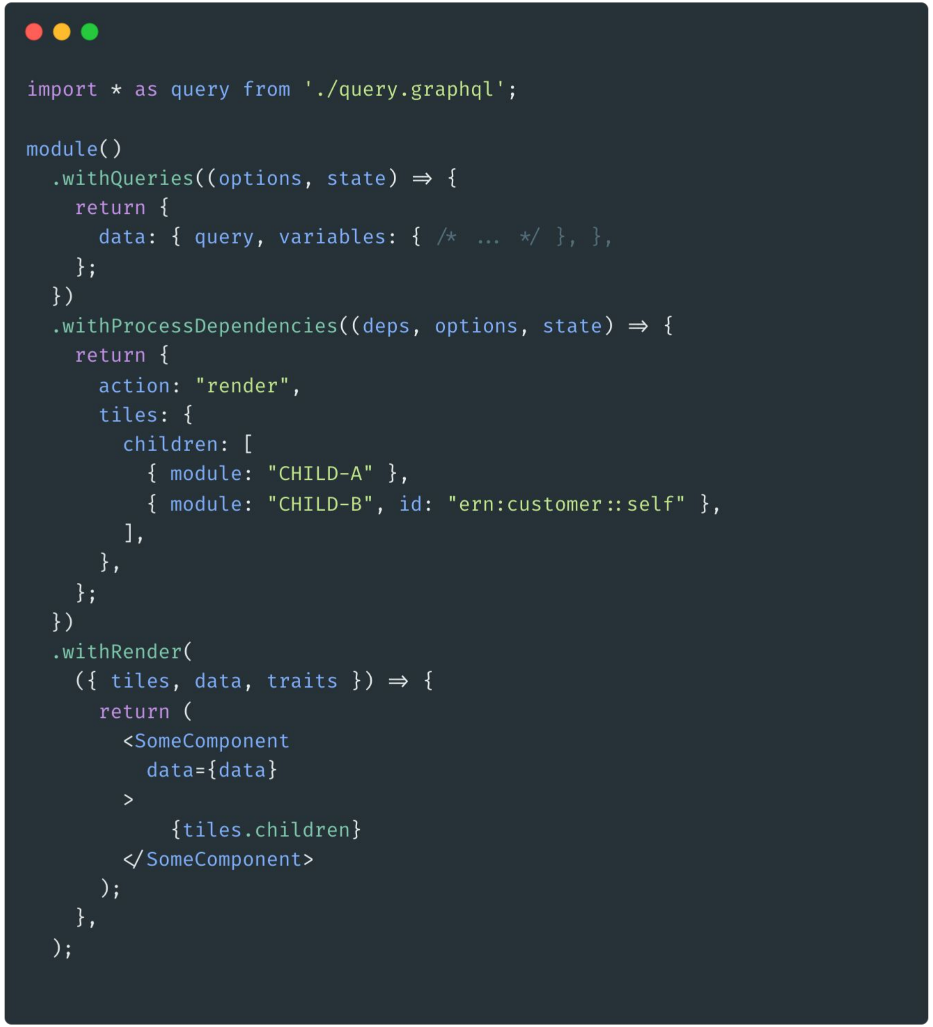
03 **Shared Concepts**
Concepts are shared across platforms to allow all engineers to work on all platforms easily

Zalando's Web Architecture

“Rendering Engine”

- In-house framework built on top of **React**
- Can be thought of as “**NextJS, but shaped for Zalando’s needs**”
- Offloads common application needs into an **opinionated API**

More details in our Blog:
>> engineering.zalando.com



```
import * as query from './query.graphql';

module()
  .withQueries((options, state) => {
    return {
      data: { query, variables: { /* ... */ }, },
    };
  })
  .withProcessDependencies((deps, options, state) => {
    return {
      action: "render",
      tiles: {
        children: [
          { module: "CHILD-A" },
          { module: "CHILD-B", id: "ern:customer::self" },
        ],
      },
    };
  })
  .withRender(
    ({ tiles, data, traits }) => {
      return (
        <SomeComponent
          data={data}
        >
            {tiles.children}
        </SomeComponent>
      );
    },
  );
```

The missing pieces

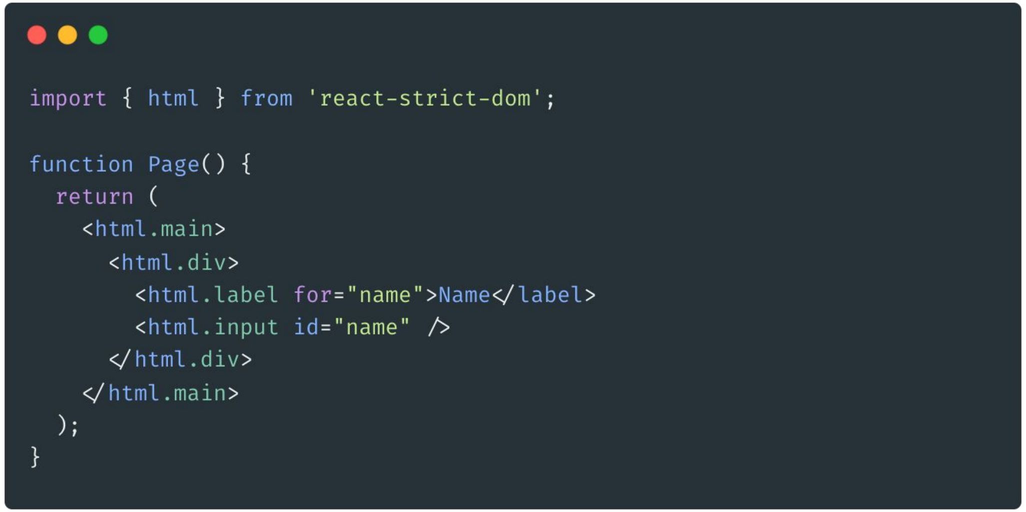
01

Rendering cross platform

While React itself is cross platform we also need components that are cross platform to be rendered.

Rendering cross platform
React Strict DOM

- Abstracts HTML like components
- Abstract CSS and adds missing RN capabilities like pseudo classes & media queries

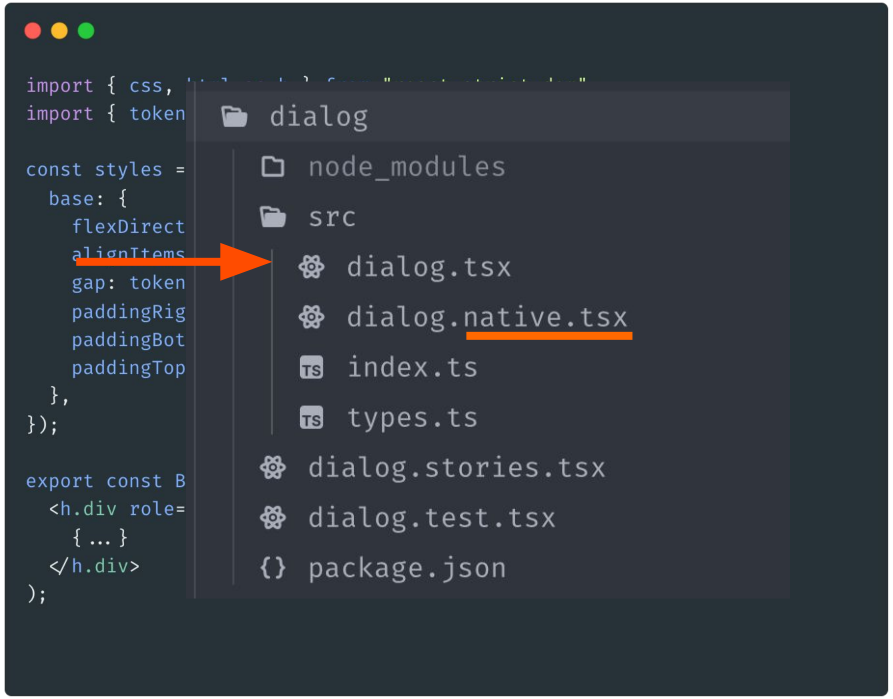


```
import { html } from 'react-strict-dom';

function Page() {
  return (
    <html.main>
      <html.div>
        <html.label for="name">Name</label>
        <html.input id="name" />
      </html.div>
    </html.main>
  );
}
```

Rendering cross platform **Component Library**

- React Strict DOM powered components
- Stylex for reusable tokens and theming



```
import { css,
import { token

const styles =
  base: {
    flexDirect
    alignItems
    gap: token
    paddingRig
    paddingBot
    paddingTop
  },
});

export const B
  <h.div role=
    { ... }
  </h.div>
);
```

```
dialog
  node_modules
  src
    dialog.tsx
    dialog.native.tsx
    index.ts
    types.ts
  dialog.stories.tsx
  dialog.test.tsx
  package.json
```

The missing pieces

01

Rendering cross platform

While React itself is cross platform we also need components that are cross platform to be rendered.

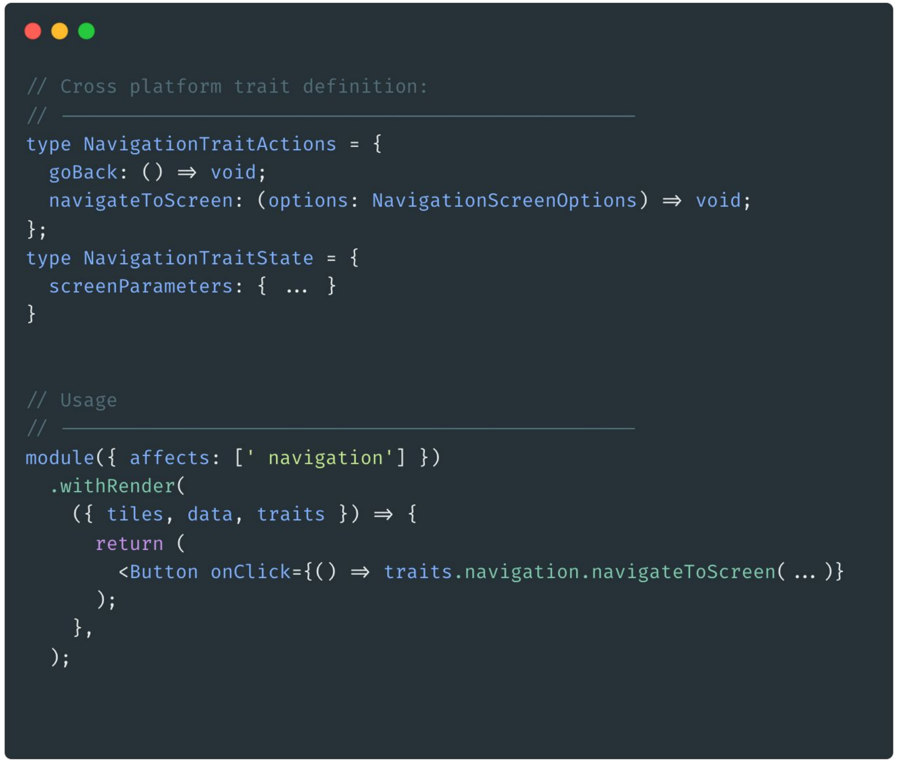
02

Abstracting common application code & state

In order the keep concepts the same across platforms we want to abstract common functionality in order to have the same APIs available across platforms despite functionality.

Abstracting common functionality
Traits

- Opinionated **state management**
- Shared **type definition** for both actions and state
- **Per platform implementation** with the option to have them shared as well



```
// Cross platform trait definition:
// ──────────────────────────────────────────────────
type NavigationTraitActions = {
  goBack: () => void;
  navigateToScreen: (options: NavigationScreenOptions) => void;
};
type NavigationTraitState = {
  screenParameters: { ... }
}


// Usage
// ──────────────────────────────────────────────────
module({ affects: [' navigation'] })
  .withRender(
    ({ tiles, data, traits }) => {
      return (
        <Button onClick={() => traits.navigation.navigateToScreen( ... )}
      );
    },
  );
```

Conclusion

01 Gradual adoption requires non-standard implementation

02 Cross platform is a double edged sword

03 React Native is a game changer

Thank you

Join the engineers
at Zalando.

zalando