DEVELOPER DIARY

JOURNEY OF A PAYMENT INTEGRATION

Tech touched and lessons learned

WHO AM I?

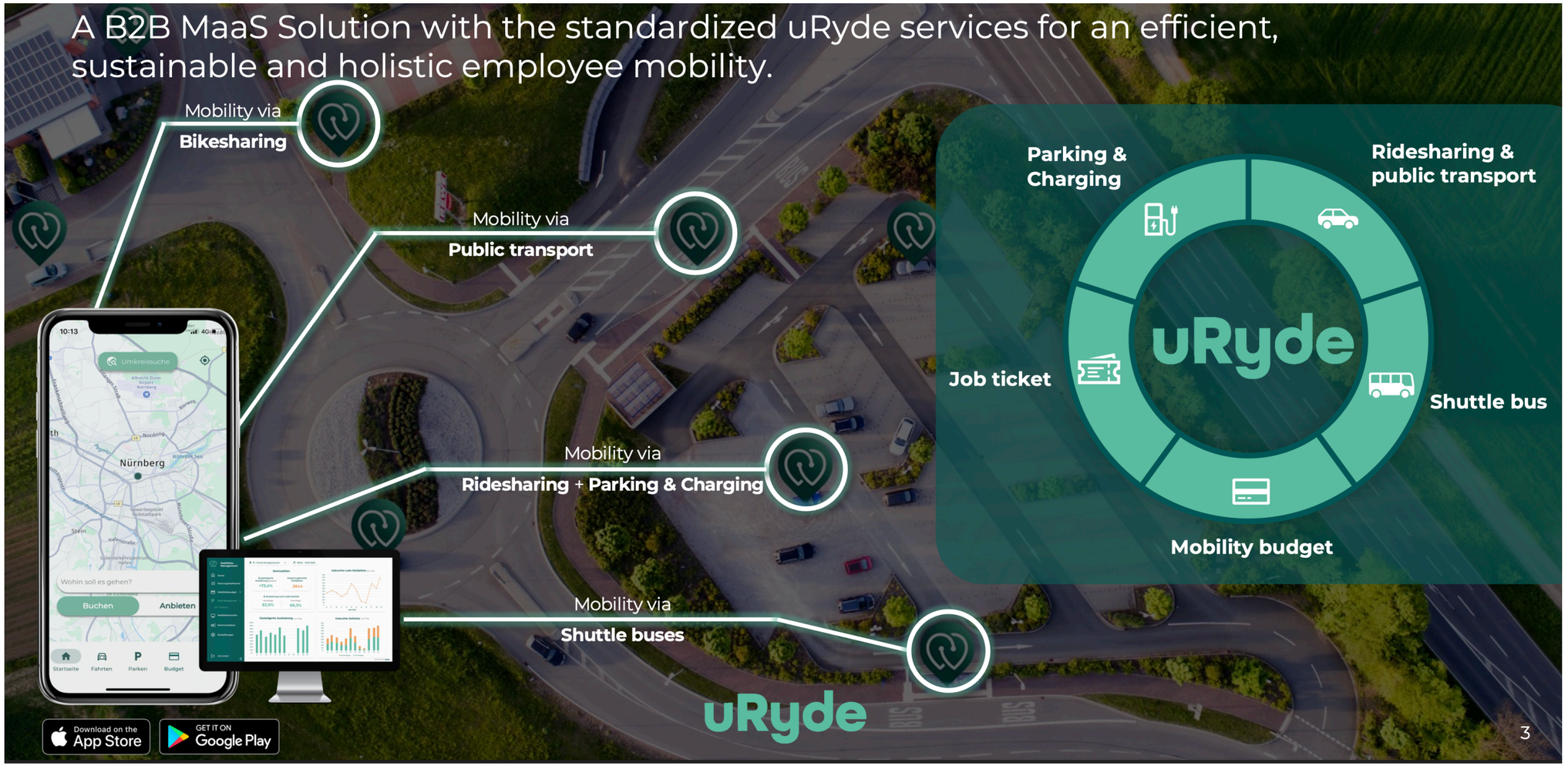
CTO & Co-Founder uRyde

Founder and Organizer GDG Nuremberg, Germany

GDG

Founder and Organizer CTO Lunch Nuremberg, Germany

WHAT IS URYDE?



DISCLAIMER

“All entities and events in this talk
- even those based on real entities and events -
are entirely fictional.
In addition, the following talk contains explicit language and should be avoided by anyone.”

DEVELOPER DIARY

WHAT IS THIS TALK ABOUT?

Dear Diary,

finally finished the web refactoring of IAintStripePayment and gotta say:

» It's crazy how many technologies can be involved in one refactoring (Tech Touched) and

» How good intentions and something that is "simple-to-integrate" can develop into a real PITA (Lessons Learned).

PITA?

LESSONS LEARNED

What's a PITA?

» Audience answers only!

» No audience answers - no talk!

TECH TOUCHED

» Flutter / Dart

» Plugins / Channels

» WebView

» In-App Browser

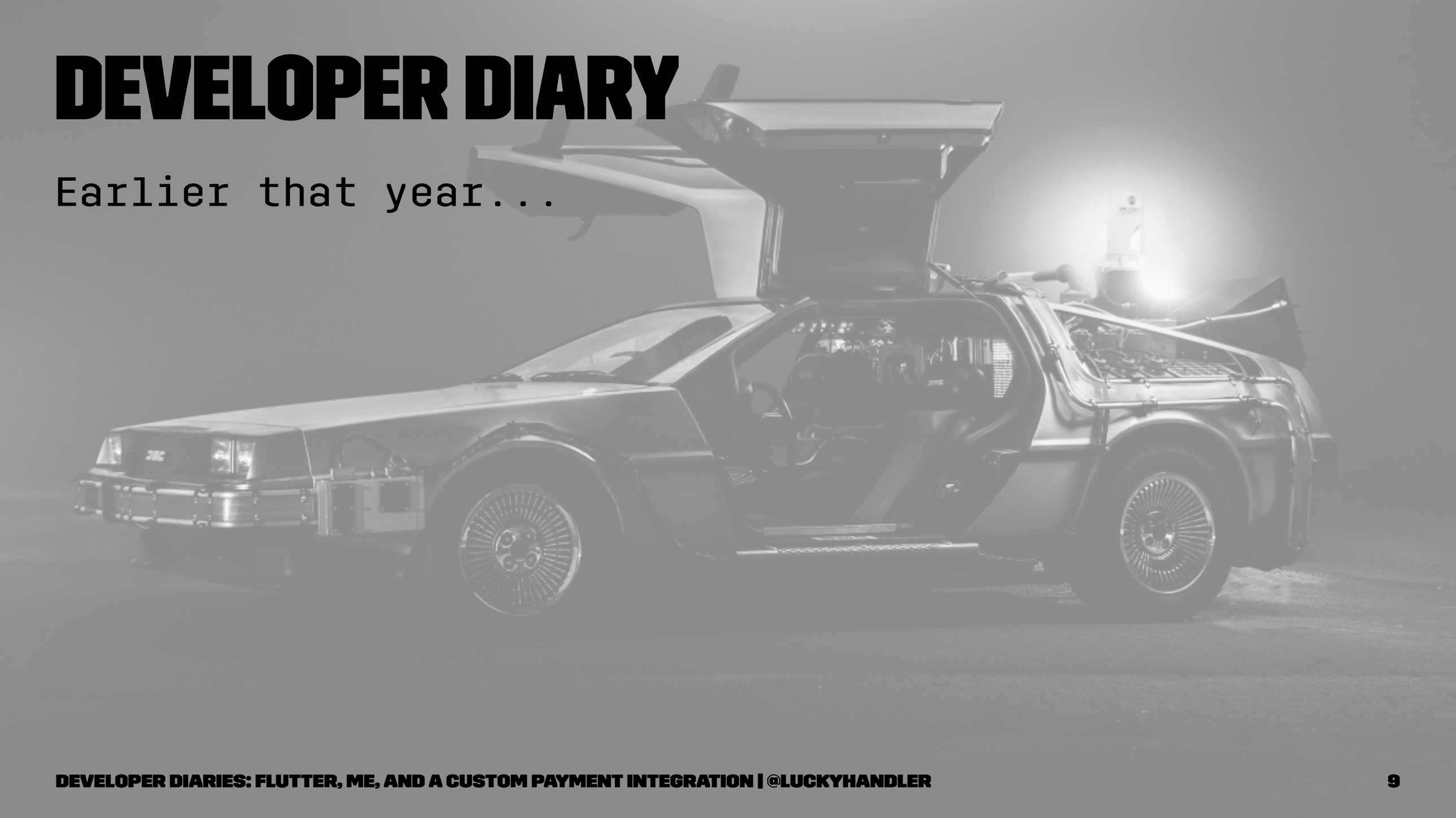
» Deep Links

» Firebase Cloud Functions

» Firebase Hosting

DEVELOPER DIARY

Earlier that year...



DEVELOPER DIARY
FIRST INTEGRATION

Dear Diary,

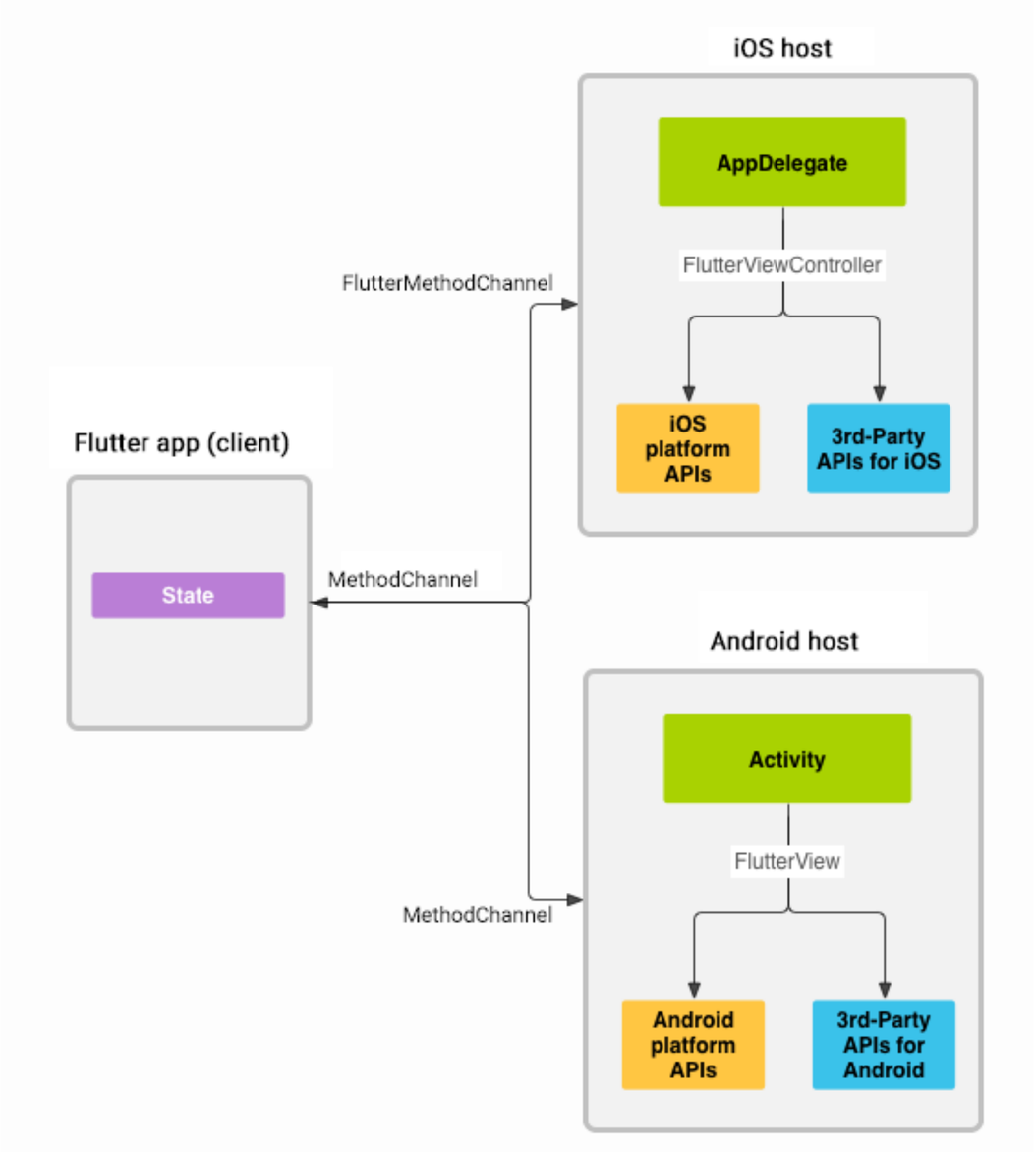
today I finished the integration of the native SDKs in Android and iOS.

It's really a pain that those libs are not distributed via maven or pod but need to be downloaded and included manually.

Still, got my hands dirty on channel communication now and thinking of making it a plugin...

SPOTLIGHT
CHANNELS

- A client sends a message to a platform via method channel (method call)
- The platform
  - receives those messages
  - does the necessary logic
  - sends back the result via the method channel



SPOTLIGHT
CHANNELS
EXAMPLE FLUTTER

```
Future<String> determineFlavor() async {
    final channel = const MethodChannel('flavor');
    final response = await channel.invokeMethod<String>('getFlavor');

    return response;
}
```

SPOTLIGHT
CHANNELS
EXAMPLE ANDROID

```
class MainActivity : FlutterActivity() {
    private val flavorChannel = FlavorChannel()

    override fun configureFlutterEngine(flutterEngine: FlutterEngine) {
        GeneratedPluginRegistrant.registerWith(flutterEngine)
        flavorChannel.configure(flutterEngine.dartExecutor.binaryMessenger)
    }
}
```

SPOTLIGHT
CHANNELS
EXAMPLE ANDROID

```
package de.uryde.app.channels.flavor

import io.flutter.plugin.common.BinaryMessenger
import io.flutter.plugin.common.MethodChannel
import de.uryde.app.BuildConfig

class FlavorChannel {
    fun configure(messenger: BinaryMessenger) {
        MethodChannel(
            messenger,
            "flavor"
        ).setMethodCallHandler { call: MethodCall, result: MethodChannel.Result ->
            if (call.method == "getFlavor") result.success(BuildConfig.FLAVOR)
            else throw NotImplementedError()
        }
    }
}
```

SPOTLIGHT
CHANNELS
EXAMPLE IOS

```swift
override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
) -> Bool {
    let controller = window.rootViewController as! FlutterViewController

    methodChannelFlavor = FlutterMethodChannel(name: "flavor", binaryMessenger: controller as! FlutterBinaryMessenger)
    methodChannelFlavor?.setMethodCallHandler { (call, result) in
        if call.method == "getFlavor" {
            let flavor = Bundle.main.infoDictionary?["Flavor"] as? String
            result(flavor)
        } else {
            result(FlutterError(code: "", message: "method not supported", details: nil))
        }
    }

    GeneratedPluginRegistrant.register(with: self)

    return super.application(application, didFinishLaunchingWithOptions: launchOptions)
}
```

SPOTLIGHT
PLATFORM PLUGIN
MAKE IT A PLUGIN

» To encapsulate the logic you can extract this channel communication into a package.

» A package that contains also native code is called a plugin.

» Find a more detailed battery example in the docs[3].

[3] https://docs.flutter.dev/platform-integration/platform-channels

DEVELOPER DIARY
FIRST PITAS

Dear Diary,

today VISA updated their logo. Logos are hardcoded in the SDKs. We therefore had to replace the SDKs in Android and iOS which was a real pain in the butt.

PROBLEMS WITH PLUGIN APPROACH

BREAKING CHANGES TIMELINE

October 2020	January 2021	February 2021	March 2021	June 2021	August 2022	October 2023
Android Embedding V2	Introduction 3D Secure	Activity Result API replaces onActivityResult	Null Safety	Major Version Payment SDK	Major Version Payment SDK	Updates to Gradle AAR Library Integration and flatDir Warnings
Changes in Android	Changes in iOS / Android	Changes in Android	Changes in Flutter / iOS & Android	Changes in iOS / Android	Changes in iOS / Android	Changes in Android?

PITAS WITH PLUGIN APPROACH
DOCUMENTATION

- Native docs are meant for native integration
  - Find out how to include code into FlutterActivity instead of default Activity
  - Find out how to include code into AppDelegate instead of default AppDelegate
- IAintStripePayment libraries not up-to-date with native platforms' developments
- Manual proprietary library integration

LESSONS LEARNED
PLUGINS

The combination of
 - Breaking Changes
 - Manual Library Integration
 - Native Docs vs Channel Integration

is / can be a PITA

DEVELOPER DIARY
A BRIGHT NEW FUTURE

Dear Diary,

we finally decided to get rid of those libs and use a simple web-based payment interface.

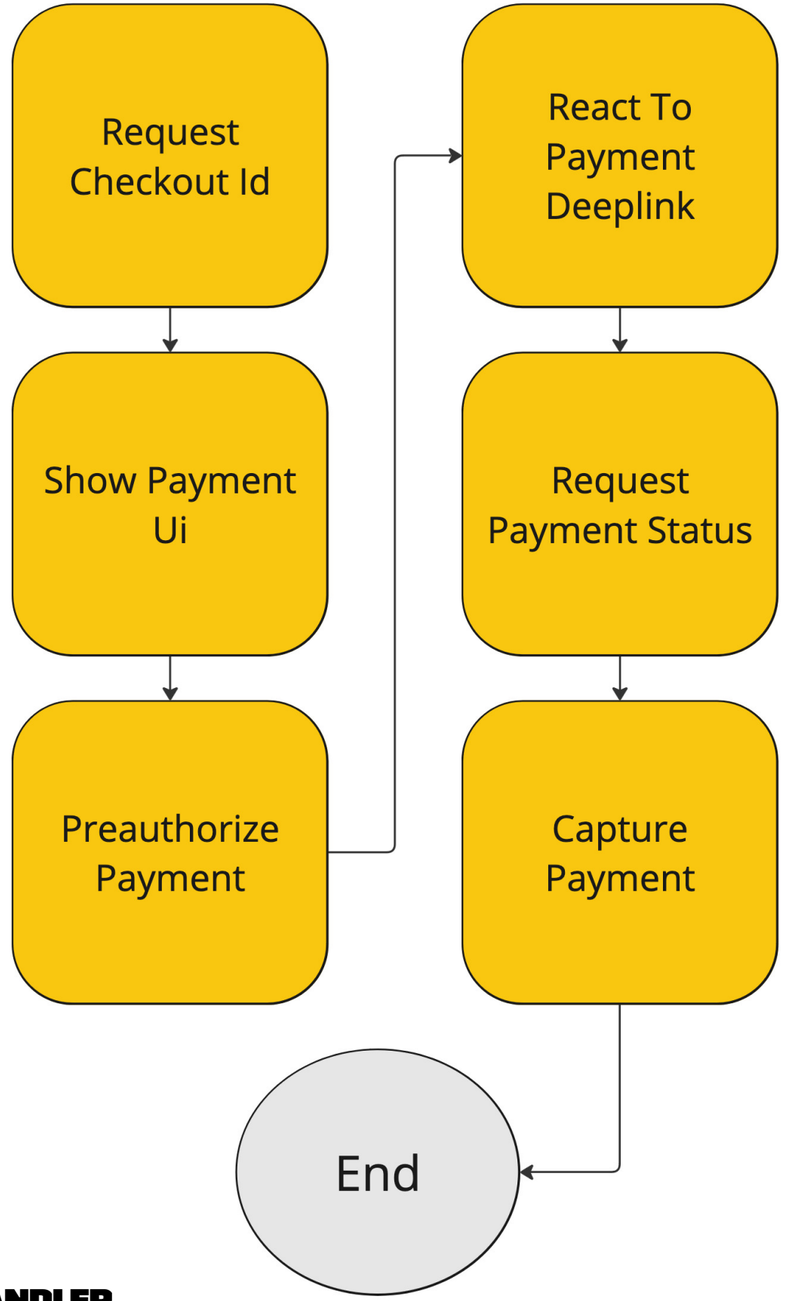
IAintStripePayment also provides a SAQ-A compliant[4] html payment-form solution which we are going to use.

[4] SAQ-A is one of several SAQ types and is designed for merchants who have outsourced all cardholder data functions to validated third parties and do not store, process, or transmit any cardholder data on their systems or premises.

DECISION TO MOVE TO WEB

WHAT TO REPLACE

The payment in general stays as it is, we only exchange the UI.



DECISION TO MOVE TO WEB

HOPE

» No breaking changes any more

» Future-proof solution

» Promise: Integration via Web easy as a pie

DECISION TO MOVE TO WEB

REALITY

- WebView too insecure
- Choose correct In-App-Browser package that intercepts on close behaviour
- Serve HTML from URL so that In-App-Browser can consume it
- Distribute Apple Pay Site Association file from this URL
- Show Google Pay in native app instead of browser
- Make sure that deep links work correctly and lead back into app

TECH TOUCHED

» Flutter / Dart ✔

» Plugins / Channels ✔

» WebView

» In-App Browser

» Deep Links

» Firebase Cloud Functions

» Firebase Hosting

WEB - 1ST TRY
PAYMENT SCREEN WIDGET WITH WEBVIEW

```
Scaffold(
    body: BlocBuilder<PaymentBloc, PaymentState>(
        builder: (context, state) {
            if (state is OnShowPaymentUi) {
              return InAppWebView(
                initialData: InAppWebViewInitialData(
                  data: state.htmlString,
                ),
                shouldOverrideUrlLoading: (controller, navigationAction) async {
                  // Intercept deeplink / final result
                },
              );
            } else {
              // Show animation
            }
        },
    ),
);
```

DEVELOPER DIARY
NOOO WEBVIEW

Dear Diary,

today I realized that my concept of integrating the payment UI with a WebView is not possible.

The banks don't trust me that I don't

» inject JavaScript, modify the DOM, intercept network requests, and more

» make sure that I never load content in an insecure context (HTTP)

In other words, they force me to use an in-app browser. Meh.

LESSONS LEARNED
WEBVIEW

Messing around with javascript and calling sensitive things in an insecure context is a PITA for the banks and in the end for the users.
So WebView is a no go when it comes to sensitive content.

WEB - 2ND TRY
PAYMENT SCREEN WIDGET WITH IN-APP BROWSER

```
if (state is OnShowPaymentUi) {
  ChromeSafariBrowser().open(
    url: WebUri.uri(state.checkoutUrl),
  );
}
```

DEVELOPER DIARY

DISTRIBUTE THE HTML FORM

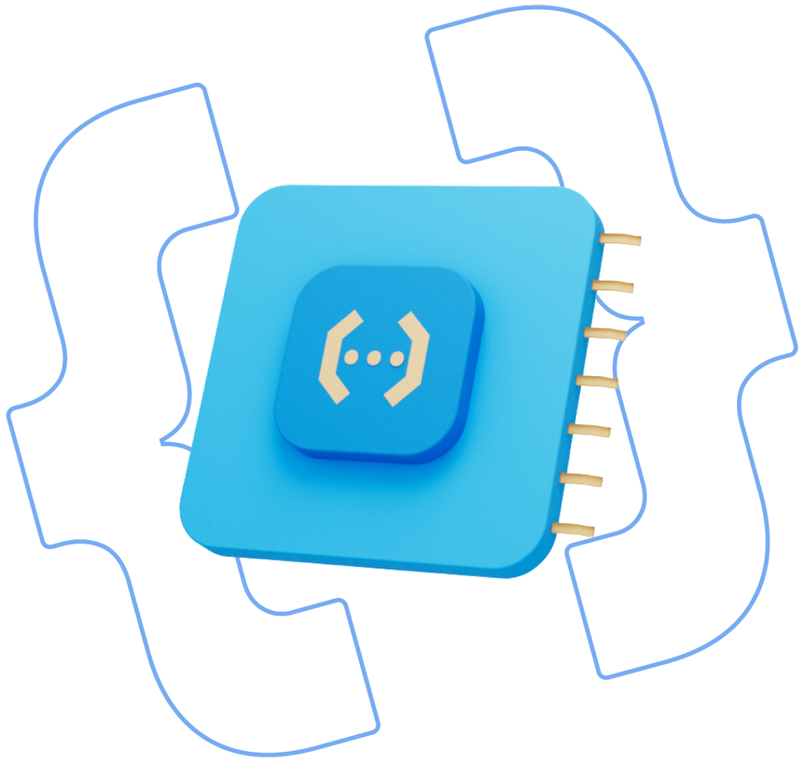
Dear Diary,

now that I need to use an in-app browser I need to distribute the html form from an endpoint.

in-app browsers cannot display simple html but only urls.
Well, that at least is easy as a pie, I'll simply write yet another cloud function!

SPOTLIGHT
FIREBASE CLOUD FUNCTIONS

“Cloud Functions for Firebase is a serverless framework that lets you automatically run backend code in response to events triggered by background events, HTTPS requests, the Admin SDK, or Cloud Scheduler jobs[1]”



[1] https://firebase.google.com/docs/functions/

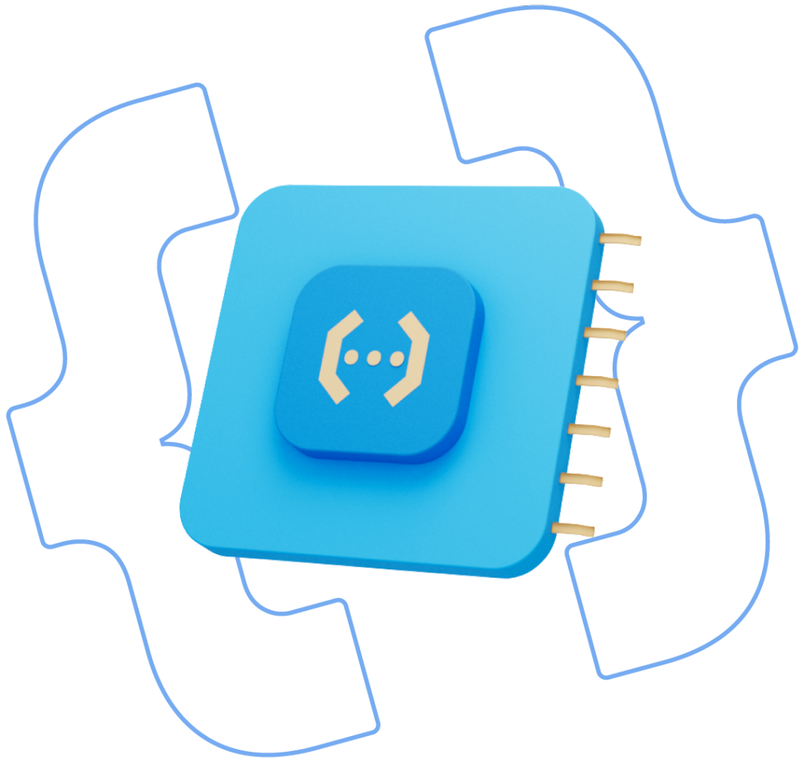
SPOTLIGHT
FIREBASE CLOUD FUNCTIONS

» Schedulers --> Run at a specific time e.g. defined by cron syntax

» Callbacks --> for Firestore Create, Update, Delete operations

» HTTPS requests

» onCall requests --> simplify authentication and security



SPOTLIGHT
FIREBASE CLOUD FUNCTIONS

```
export const checkoutUrl = functions
  .region('europe-west3')
  .https
  .onRequest(async (req, res) => {
    const checkoutHtml = buildCheckoutHtmlString({ params: req.query });
    return res.status(200).send(checkoutHtml);
  });
```

SPOTLIGHT
FIREBASE CLOUD FUNCTIONS

```
export const checkoutUrl = functions
  .region('europe-west3')
  .https
  .onRequest(async (req, res) => {
    const tokenId = req.get('Authorization')?.split('Bearer ')[1] ?? '';
    admin.auth().verifyIdToken(tokenId)
      .then((_) => {
        const checkoutHtml = buildCheckoutHtmlString({ params: req.query });
        return res.status(200).send(checkoutHtml);
      })
      .catch((err) => res.status(401).send(err));
  });
```

WEB - 2ND TRY
PAYMENT SCREEN WIDGET WITH IN-APP BROWSER

```
if (state is OnShowPaymentUi) {
  ChromeSafariBrowser().open(
    url: WebUri.uri(state.checkoutUrl),
  );
}
```

state.checkoutUrl equals
https://europe-west3-
<project_name>.cloudfunctions.net/checkoutUrl

LESSONS LEARNED
CLOUD FUNCTIONS

» Cloud Functions for Firebase lets you easily setup an https endpoint for your app.

» It's easy as a pie to even build a small web application which serves your app.

» Find our more about how to deploy and write functions in the docs.

DEVELOPER DIARY

WHICH IN-APP BROWSER PLUGIN?

Dear Diary,

I'm currently looking into different in-app browser plugins. The user might simply close the browser without paying. So either I handle this case with the widget lifecycle or the plugin comes with an onCLose callback.
Also I definitely want to use a well-maintained package.

SPOTLIGHT
IN-APP BROWSERS

FUNCTIONALITY	URL LAUNCHER	FLUTTER INAPPWEBVIEW	FLUTTER CUSTOM TABS
Manually Close Open Browser	yes	yes	yes
On Close Callback	no	yes	no
Package Size	small	big	small
Platform Support	Android, iOS, Web	Android, iOS	Android, iOS
pub.dev rating	150/ 160	150/ 160	150/ 160
pub.dev likes	7226	2284	220

LESSONS LEARNED
IN-APP BROWSERS

» When not having any special requirements simply go for the urlLauncher package.

» Don't write your code in a way that you need to close a screen when the browser closes.

» Different in-app browser plugins behave differently in certain cases. Make sure to also test with other default browsers.

DEVELOPER DIARY
PROBLEMS WITH GOOGLE PAY

Dear Diary,

I realized that when I pass the payment provider a custom url scheme the native Google Pay sheet doesn't show up but the browser.
This seems to be an undocumented requirement of IAintStripePayment.
So I need to use Universal Links in iOS and setup App Links for Android. I have to tell you, the whole integration becomes a real PITA.

SPOTLIGHT
DEEP LINKING

There are two kinds of deep links

PLATFORM	HTTP / HTTPS	CUSTOM SCHEME
iOS	Universal Link	Custom Url Scheme
Android	Web Link (deprecated) / Android App Link	Deep Link
	https://uryde.de/payment	de.uryde.app://payment

Android App Links | iOS Universal Links use HTTP URLs that link to a website domain you own, so no other app can use your links.

LESSONS LEARNED
DEEP LINKING

- A custom url scheme / deep link can be intercepted by other apps and is therefore considered insecure
- Using Universal Links / App Links in contrary is considered secure as you proof that you own the domain you are calling with
  - an Apple App Site Association (AASA) File (iOS)
  - Digital Asset Links (DAL) File (Android)

DEVELOPER DIARY
OWN YOUR STUFF

Dear Diary,

For the setup of App Links and Universal Links I need to provide documents at the .well-known path of my domain.

Loving how easy it is to setup a Firebase Hosting project for my domain and also host those files there.

Besides, I can also provide the Apple Pay Certificate in the same way!

SPOTLIGHT
FIREBASE HOSTING

- With Firebase Hosting and DNS provider you can define custom domains, e.g. payment.uryde.de
- You can also host files in this hosting project, which are necessary for
  - setting up Android App Links
  - setting up iOS Universal Links
  - setting up Apple Pay
- You can as well use a rewrite rule to redirect a url to your specific cloud function

SPOTLIGHT
FIREBASE HOSTING

A rewrite rule is different than a redirect because it doesn't change the url and doesn't tell the browser to reload the new url. Also, it is useful for Apple Pay which needs an offically owned domain for payment

So moving

» https://europe-west3-<project_name>.cloudfunctions.net/checkoutHtml to

» https://payment.uryde.de/checkout

is no problem with the help of the rewrite rule

SPOTLIGHT
FIREBASE HOSTING

```
{
  "hosting": {
    "site": "uryde-payment",
    "public": "public",
    "rewrites": [
      {
        "source": "/checkout",
        "function": {
          "functionId": "checkoutHtml",
        }
      }
    ],
    "headers": [
      {
        "source": "/.well-known/assetlinks.json",
        "headers": [
          {
            "key": "Content-Type",
            "value": "application/json"
          }
        ]
      }
    ]
  }
}
```

LESSONS LEARNED
FIREBASE HOSTING

- It's super-easy to quickly setup a hosting project
- Use hosting for provising necessary files for your subdomains
  - for Deep Linking
  - for Payment
- Use hosting's rewrite rules to build a well defined API

TECH TOUCHED

» Flutter / Dart ✔

» Plugins / Channels ✔

» WebView ✔

» In-App Browser ✔

» Deep Links ✔

» Firebase Cloud Functions ✔

» Firebase Hosting ✔

DEVELOPER DIARY
FINAL WORDS

Dear Diary,

Finally finished the web integration. It's been way more work than expected but hopefully now without breaking changes!
But honestly, I would have preferred to stay with the plugin approach in the end.

Considering refactoring to plugin approach soon!

LESSONS LEARNED

- When choosing a 3rd party provider also consider
  - Development / implementation costs
  - Good documentation
- Don't be afraid of writing plugins / opening channels.
- When going for web based things (e.g. also login) be careful:
  - WebView is not secure.
  - In-App Browsers are a PITA in certain use cases.
- Be creative with your tech.

THANK YOU!

» Find me on LinkedIn

» Find me on X and Mastodon as @luckyhandler