Session title	Description	受講対象者 / Intended Audience	Session format	Language	カテゴリ / Category	スピーカー / Speakers
Prepare your project for KMM Will you take the leap of faith?	Kotlin Multiplatform is not yet stable. Despite this, many companies have taken the leap of faith and have started adopting it in their production projects. In this session, we will see an overview about KMM (Kotlin Multiplatform Mobile) and how to prepare your project to adopt it. A deep look at the modularization architecture and how it facilitates sharing the maximum layers between the different stacks. We will make the focus on Android and iOS, specially coming from native projects. We will explain how to migrate your project in order to integrate it with a KMM module. Lastly, we can not forget that we can use the shared modules on Desktop and Web Apps.	Kotlin Basic MVVM architecture KMM adption starting from Android Project Modularisation	40 minutes	English	Kotlin	Oussama Ben wafi
Android/Chromeで体験できる認証のための標準化仕様の現在と未来	パスワード認証の限界が語られ、"パスワードレス" という用語が使われ始めてから数年が経ちました。 実際に、我々の周りにあるユーザー認証の仕組みはどう変わったでしょうか？パスワードは完全になくなりましたか？ 残念ながら、パスワード認証はまだたくさんのサービスで使われています。 しかし、2段階認証はGoogleが本格的にコンシューマ向けに始めてから10年以上経ってようやく一般的になりつつあります。 パスワード認証の後にSMSやEメールで短い文字列を送信したり、Google Authenticatorが表示した文字列を利用しているサービスはたくさんあります。 FIDOはどうでしょう。指紋認証などでモバイルアプリやWebアプリにログインできるサービスも少しずつ出てきています。 また、GoogleやAppleのような巨大プラットフォーマーが提供するID連携機能を利用するアプリも少しずつ増えています。 このような様々な認証機能は、Web標準規格やブラウザのAPI実装によって実現されています。 これらの仕組みですが、今後はどう進化していくと思われますか？ 直感的に想像できることといえば「手元のスマホを使ってログイン」することでしょう。 これはいわゆる "Decoupled Authentication" という考えであり、認証処理を行う端末とその結果を利用するサービスが動作する端末が分離することで様々なユースケースへの適用が考えられます。 本セッションでは * WebOTP * OATH(TOTP) * FIDO, WebAuthn, passkey * Identity Federation & FedCM といったキーワードについて、 * AndroidのChrome環境でどう動くのか * PCのWebアプリケーションとAndroid端末との組み合わせで何が実現できるのか といったあたりを、デモ動画等を交えて解説します。 * 個別の仕様はどういうもので、現在はどのように使われているのか * 将来はどうなる(ことを夢見ている)のか という、現在と未来の姿を想像できることを目的とします。本セッションにより、少しでも世の中のサービスの認証機能が安全安心、利便性に寄与したものになるきっかけとなれば幸いです。	以下のような開発者が対象となるでしょう。 1. モバイルアプリからブラウザを呼び出して認証処理やID連携を実装している開発者 2. AndroidのChrome上で動作することを意識しているWebアプリケーション開発者 3. PC向けのWebアプリケーションとAndroid端末を組み合わせた認証機能に興味がある開発者 4. Web標準技術が今後どのような姿になり、使われていくかに興味を持っている開発者	40 minutes	日本語 / Japanese	Security / Identity / Privacy	ritou
The world of Android wireless communications without Internet	Since smartphones in recent years are usually connected to the Internet in some way, people are usually not so aware that the device you hold in your hand is equipped with a wealth of communication methods that are not dependent on it. In fact, Android officially supports Bluetooth (Classic), Wi-Fi Direct, and Bluetooth Low Energy (BLE), and most devices available today support these APIs. This session will provide an overview of these features, a brief description of how to use them, and general guidelines on how to use each one differently. It is hard to keep track of all of these at once, so this session should be interesting for Android developers who have not used many communication methods other than the Internet.	This session is a 101 session on Bluetooth (both classic and BLE) and Wi-Fi Direct available for Android, so interested Android developers can watch without prior knowledge. On the other hand, viewers who are already familiar with these in their work may find them slightly lacking.	40 minutes	English	Androidプラットフォーム (Android Platform)	@fushiroyama
FlutterをROS(ロボティクス)と一緒使ったらマジですごかった話	通常、RoboticsではROSとAndroidまたはQTを組み合わせてユーザーアプリケーションを開発しています。今回Flutterが発表され、デスクトップ、組み込み領域で使用できるように機能が拡張されました。生産性が高いユーザーアプリケーションを作成するためにロボットにFlutterを適用してみました。 そのためには、ロボットとアプリ間通信するためのプロトコルライブラリの開発が必要です。パッケージ開発方法と物語、そして実務で使用している例示を通じて使用機を紹介したいと思います。 関連技術 - Flutter & Flutter Desktop (Linux) - ROS2 - Websocket - gRPC	- FlutterやFlutter desktopに興味がある人 - Androidとrosを使ってみた人 - ロボットが好きな人	40 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	DREAMWALKER
The art of hiding sensitive info in plain sight	An android application usually holds sensitive information, like names, tokens and even sometime bank data that we would NOT like to fall into the wrong hands. These days applications can be reverse engineered, and reveal a great chunk of their code. In this session, we will find out exactly which parts of our code can be hacked, and we will also look at ways of securing our data.... in plain sight!	No specific audience. Just general Android knowledge is needed and interest about security.	40 minutes	English	Security / Identity / Privacy	George Kortsaridis
Getting started with Dagger and Hilt	Dagger is a powerful tool for setting up dependency injection for your app. With that comes a step learning curve, this talk will demystify setting up Dagger with a focus on using the Hilt library. You can also expect to learn about best practices and tips for reading error messages in Dagger.	Expected understanding of basic Android knowledge (what an activity, fragment and application are.)	40 minutes	English	開発ツール＆サービス (Productivity and Tools, Service)	Zach Westlake
アプリ アーキテクチャ ガイドの解説	最近弊社ではAndroidアプリ開発者の皆さまにより役立つ知識を与えるため、アプリアーキテクチャの設計の実装に関するガイドを公開しています。今回のセッションではこのガイドをよく理解し、より効果的に応用するための知識を共有したいと思います。 目次: - Googleのアーキテクチャ ガイドが言及しないトピック (例: Clean Architecture) - UI層 - ViewModel実装のあるべき姿とアンチパターン - ドメイン層 - あなたのアプリに役に立つ設計戦略 - データー層 - 実装の時考慮して欲しい点 - 依存性注入 - Hiltを使うべき理由とその限界 - マルチモジュール - 実際にどうわけたらいいのか	Android フレームワークの基本について熟知している方	40 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Saryong Kang
Androidでもドラッグ&ドロップがしたい！	iOSでは比較的馴染みのあるドラッグ &ドロップ操作ですが、Androidではほとんど実装を見かけません。 しかし、RecyclerViewのItemTouchHelperを利用すればリストのドラッグ &ドロップ操作を実装可能です。 また、2022年のGoogle I/OでJetpack DragAndDrop ライブラリの1.0.0が登場し、アプリ内や異なるアプリ間でのドラッグ &ドロップ操作をサポートしました。 このセッションではAndroidアプリでのドラッグ &ドロップ操作の可能性を探ります。 - ItemTouchHelperを用いたリストのドラッグ &ドロップ操作 - Jetpack DragAndDropについて	- Androidでドラッグ &ドロップ操作を実装してみたい方	25 minutes	日本語 / Japanese	Jetpack	m.coder
Offline first Flutter applications	Even though the world is becoming more "online" every day, there are still a lot of parts of the world that we all suffer connectivity problems. And, as we all know, as good developers, we should not let that affect the users' experiences in our applications. Thinking "offline" first is a concept that we all suffer to achieve. Luckily there are some tools provided to us to make our lives easier. One of them is DataStore from AWS Amplify. Amplify DataStore provides a programming model for leveraging shared and distributed data without writing additional code for offline and online scenarios, which makes working with distributed, cross-user data just as simple as working with local-only data. With this talk, I will share my experience with thinking "offline" first with the applications I worked on before. You will learn about, what Flutter offers for being "offline" first, how you can achieve a great experience for your users by architecting your app the correct way, and most importantly, you will learn what NOT to do. :)	Intermediate knowledge on programming and beginner knowledge on Flutter is important	40 minutes	English	クロスプラットフォーム (Cross-platform Development)	Salih Guler
実践 `edge-to-edge`: 端から端まで解説	`edge-to-edge`とはその名の通り、システムの領域も含めた画面の "端から端" を全てアプリの領域として描画する機能です。 Googleはアプリへの没入感を高めるために、この`edge-to-edge`を推奨しています。 `edge-to-edge` を適用するには、アプリの描画が全画面になるようにフラグを設定し、システムバー（ステータスバーとナビゲーションバー）の色を設定するだけでも十分に思われます。 しかし、実際には考慮すべき点や落とし穴があるため一筋縄では適用できません。例えば、既存のアプリに適用する場合は、UIデザインの変更が必要になります。 他にも、Android デバイスによっては、システムバーに対してデバイス固有の変更が施されている場合があり、その対処も簡単ではありません。 本セッションでは、実際にコミュニケーションアプリLINE に`edge-to-edge`を適用した際に得られた知見を元に、以下の内容を紹介します。 ・Googleが提唱する`edge-to-edge`の適用ポリシーについて ・UIをデザインする上での`edge-to-edge`の適用基準について ・通常のアプリとJetpack Composeで構成されたアプリそれぞれに対しての適用方法 ・LINE Androidアプリに対して`edge-to-edge`を適用する際に発生した問題点・落とし穴やその解決方法について ぜひ皆さんもこのセッションから`edge-to-edge`の知見を得て、アプリへの適用を実践してみてください！	・既存のAndroidアプリに`edge-to-edge`に適用したい人 ・Android OSの新機能を既存のアプリに適用したい人	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Yuki Ando

Molecule: Using Compose for Presentation Logic	Jetpack Compose can be used for more than just emitting a user interface. Molecule is a library that allows `Flow` or `StateFlow` streams to be built using Compose. This talk will cover: - Comparing Compose to Rx and Flow APIs for presentation logic - The benefits Compose can have on readability - How Cash App uses Molecule to build presenters - How to adopt Molecule in existing apps - Testing strategies, gotchas, and other interesting quirks	Android developers who have built at least a few screens in Compose UI, and have a rough understanding of coroutines and Flow.	25 minutes	English	Jetpack Compose	Chris Horner

Add-to-Appの戦い方	AndroidとiOSに新機能を追加したいものの、WebViewでは少し用途にあわない、といった場面に出くわすことはないでしょうか。
	Kotlin+SwiftでUIを個別に実装はできるが、しかし工数的に厳しい……とき、FlutterのAdd-to-Appが利用できるかもしれません。
	
	本セッションでは、FlutterのAdd-to-Appを活用した機能開発と、そのメリット・デメリットを紹介します。
	Add-to-AppはFlutterの機能を、KotlinやSwiftから呼び出し、既存のアプリケーションに組み込むことができる機能です。
	Flutterへの段階的なリプレースのため、開発工数削減のためなど、さまざまな目的で利用できます。
	
	Add-to-Appのはじめ方は、非常に簡単です。
	しかし、Add-to-Appを採用した後に気にすべきことや、Add-to-App固有の問題などはあまり紹介されていません。
	このセッションでは、Add-to-Appを利用したアプリケーション開発について、つまりAdd-to-Appの戦い方について紹介します。
	
	* Add-to-Appとは
	  * Add-to-Appでできること
	  * Add-to-Appの仕組み
	* Add-to-Appのメリット
	  * ロジックの共通化、工数の削減
	  * 段階的なリプレース
	* Add-to-Appのデメリット
	  * アプリサイズ、ビルド時間の増加問題
	  * ビルドシステムに関わる問題
	  * スクリーンサイズ、回転問題
		* 複数プラットフォームに対応する工数に危機感を感じているエンジニア
		* 段階的なFlutterへのリプレース方法を探しているエンジニア
		* Add-to-Appを導入したものの、課題感を感じているエンジニア
			25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	koji-1009

How to Test Your Compose UI	While Compose is easy to adopt, not creating legacy code right at the start of such a journey requires some extra planning, awareness, and most importantly, testing. We'll have a look at how we can test pure Compose UIs as well as hybrid ones that have Views and composables too. In this talk, we'll learn what the semantics tree is, what is its relation to the composition, how we can manipulate it in composables using the Semantics modifier, how we can implement composables with testability in mind, and how we can test pure Compose and hybrid UIs.	Those Android developers, who are already somewhat familiar with Jetpack Compose, and want to write UI tests for the Compose-based UIs they are creating.	40 minutes	English	Jetpack Compose	István Juhos

Anatomy of Dynamic color	Material 3で登場したDynamic colorは、ユーザーの壁紙をもとにパーソナライズされたカラーシステムをアプリに適用できる機能です。色という非常に主観的な要素をカスタマイズ可能にすることで、ユーザーの好みやアクセシビリティなどのニーズにアプリのUIを対応させることができます。
	
	Android 12以上を対象とした機能ですが、今後の対象OSの使用率上昇に伴い、よりユーザーに広く使われる機能となっていくことが予想されます。
	
	しかし、いざDynamic colorのアプリへの導入を考える際に、自動生成されるカラーシステムは信頼できるものなのか、プロダクト固有の意味ある色まで動的に変更されてしまうのかなど、さまざまな疑問が湧いてくるかと思います。
	
	そこで本セッションでは、Dynamic colorはどのような仕組みの上に堅牢なカラーシステムを構築し、そしてアプリ固有の色とどのように向き合っているのかについてお話します。その際に、Dynamic colorについて内部の仕組みから徹底解剖することで、聴講者の皆様がDynamic colorを深く理解し、導入についての議論をチームで円滑に進められるようになることを目指します。
	
	アジェンダ
	- Dynamic colorの概要
	- Dynamic colorにおける色空間
	  - 従来のHSL色空間の課題
	  - HCT色空間の採用
	- Accessibility要件を満たすカラーパレットの動的な構築
	- 調和の取れたCustom colorsの導入
		- Dynamic colorを理解したい方
		- Dynamic colorの導入を検討している方
			25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	ロクネム

Jetpack Composeを用いて、Canvasを直接触るようなコンポーネントを作成する方法	Jetpack Composeは、ライブラリにあるコンポーネントでアプリを組むのは従来のAndroid Viewに比べて簡単で開発時間の短縮にもつながる優れたUIツールキットです。 一方、Android View時代だったら、Viewを継承して、オーバーライドしたonDraw()に処理を ガリガリ書いていたであろう完全にオリジナルコンポーネントを作成しようとした時、どうしたらよいのか悩まれる方もいるのではないでしょうか。 私は去年 Jetpack Composeを用いて、そのようなコンポーネントを持つアプリをリリースしました。 ユーザーから受け取った端末内画像を表示し、その上を指でなぞると軌跡が切り取られて透明表示のレイヤーが表示されるようなアプリです。 そのアプリで使用しているコードを例に、Jetpack Composeを用いて完全なオリジナルコンポーネントを作成する際にヒントにできるような内容をお話しする予定です。	・Jetpack Composeを用いて、Canvasを直接触るようなコンポーネントを作成する方法に興味がある方 ・Jetpack Compose初心者の方	40 minutes	日本語 / Japanese	Jetpack Compose	kkkkan
Jetpack Compose で Material Design 3	Now in Android アプリなど最近公開されたサンプルアプリでは Material Design 3 の Compose ライブラリ (androidx.compose.material3) が利用されています。 現状ではまだ alpha ですが、主要なコンポーネントが一通り揃ってきており、Material Design 3 と合わせてキャッチアップするいいタイミングです。 このセッションではまず Material Design 3 について復習し、次に compose.material3 でのテーマの作り方を解説します。 Material Design 3 のテーマを自分のアプリ用にカスタマイズするには、Color, Shape, Typography などの設定がどのコンポーネントのどこで使われているかを把握する必要があります。 これらの利用箇所を整理した上で compose.material3 に用意されているコンポーネントを紹介します。 最後にアプリのブランドカラーを保持しつつ Dynamic Color に対応する方法を紹介します。	Jetpack Compose を使っていて Material Design 3 への移行を考えている人 Material Design 3 について知りたい人	40 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Yuki Anzai
Android/iOSアプリを協調開発するチーム ～スクラム開発の実践とその先へ～	Android・iOSエンジニアが1つのアプリをそれぞれKotlinとSwiftで作る際、どうすれば効率良く開発を進めることができでしょうか？異なるプラットフォーム開発の間でどのようにして知見を共有すれば良いのでしょうか？この セッションでは、Androidエンジニアと iOSエンジニアが協調しながら開発をしている例の一つとして、日々進化している私達のチームを紹介します。 私達はGoogleが提唱するDesgin Docsと呼ばれるドキュメント手法を、マルチプラットフォーム用に拡張した手法を提案します。Desgin Docsは実際に実装を行う前に書くドキュメントの１つで、開発する機能の背景・設計・検討する実装方法をまとめ、レビュープロセスを経ることで属人性や実装の出戻りを排除します。私達はAndroid・iOSで共有のDesign Docsを用意することで、マルチプラットフォーム開発であってもAndroidチームとiOSチームの知見を共有する試みを行っています。 また、スクラム開発を実践するだけではなく我々に合わせて様々なチャレンジをしてきました。例えば、より使いやすいアプリをお客様に使ってもらうため、ドッグフーディングと呼ばれる、QAとは別に自分達で成果物を触って改善や問題を事前に発見する手法をスクラムイベントの１つとして日常的に実践しています。 プレゼンテーションでは、Android/iOSエンジニアが一つのチームとして楽しく協調しながら開発を進めていく経験や工夫について、模擬機能開発を例にお話します。	Android/iOS向けアプリをチームで開発している人。 Android/iOSエンジニアが楽しく協力しあって開発したい人	25 minutes	日本語 / Japanese	開発体制 (Development Process)	Kuu, otao
詳解Google Playの新しい定期購入 ~オファーの活用や実装例を添えて~	Google Playの従来の定期購入では、何百ものSKUを作成したり、SKUを作成するたびにアプリを更新しないといけないなど、柔軟性に欠けたものとなっていました。 そこでGoogleは、Google Playの定期購入の販売方法を再構築したとGoogle I/O 2022で発表しました。 定期購入の機能や期間ごとにSKUを作成しなければならなかった仕組みを再構築し、定期購入に対して期間が紐づく基本プランを、基本プランに対してオファーを提供できるようにしたことで、定期購入のライフサイクル全体でオファーを作成する柔軟性を提供できるようになりました。 しかし、既に提供している定期購入では新しい定期購入の提供の形になっているわけではなく、従来の仕組みのまま新しい定期購入に自動移行されています。 新しい定期購入の仕組みを最大限活用するには、従来の定期購入と新しい定期購入の違いや、どのようなオファーを提供できるのかをチーム全体で理解した上でオファーを考える必要があります。 本セッションでは、Google Playの新しい定期購入の仕組み やオファーの活用方法、実際の実装についても詳しく解説し、新しい定期購入をより活用するための手助けをします。 発表予定の目次 - 従来の定期購入とその課題について - 再構築された新しい定期購入 - 自動移行された既存の定 期購入 - 新しい定期購入の作り方 - 既存の定期購入における新しいオファーの活用方法 - Play Billing Library 4.xから5.xへの移行手順 - オファーの選択ロジックの決め方と実装方法	Google Playで定期購入を提供している人 Google Playでこれから定期購入を提供したい人 Google Playの定期購入の仕組みや実装に興味がある人 Google Playの新しい定期購入でオファーを活用したい人 Play Billing Libraryの最新情報について知りたい人	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	syarihu

Context Receiversに思いを馳せる	Kotlin 1.6.20より、Context Receiversがfirst previewとしてリリースされました。今まで非常に堅実に言語仕様を積み上げてきたKotlinですが、Context ReceiversによってKotlinでのコーディングに大きな変化が訪れる…かもしれません。もちろん、今までの通りの記述は可能ですし、自由度の高い言語仕様は濫用すべきではありません。Kotlinという言語を好んで選んでいる皆さんであれば重々承知の事でしょう。私もその1人です。 しかし、妄想は自由です。 Context Receiversの現在までの道のりを振り返りながら、Context ReceiversがKotlinのコードにどのような変化をもたらすか、どんな便利なExtensionが用意できそうか、Context Receiversが解決したいものは何なのか、…このセッションがそういった事を考えるきっかけとなれば幸いです。	KotlinのExtensionsの仕様に興味がある KotlinのContext Receiversの仕様に興味がある Kotlinでのコーディングの引き出しを増やしたい	25 minutes	日本語 / Japanese	Kotlin	uzzu
Camera Xライブラリの魅力と最新機能を紐解く	Camera Xライブラリは、カメラアプリの開発を容易にすることを目的としたJetpackライブラリです。Android 5.0以降を搭載したデバイスをサポートしており、Camera 2ライブラリと比較して圧倒的に少ないコード量で一般的なカメラのユースケースや、デバイスのネイティブカメラアプリと同じ機能を実現することができます。 最新のstable版であるバージョン1.1.0は2022年6月末に公開されました。実に1年ぶりとなるマイナーバージョンのアップデートであり多くの新機能が加わっています。 本セッションでは、一見複雑多岐に見えるCamera Xライブラリを分かりやすく紹介しつつ、バージョン1.1.0の新機能を始めとした様々な魅力をお伝えすることで、カメラ機能を生かした新たなアイデアを生み出すきっかけとなることを目指します。 発表予定の目次 ・Camera Xライブラリの概要、基本的な実装の解説 ・バージョン1.1.0での新機能の解説・実演 ・バージョン1.1.0ではまだ実現できず課題を抱えていることの紹介	・カメラ機能をアプリに組み込みたいと考えている人 ・現在Camera 2ライブラリなどを使っている人 ・Camera Xライブラリを使うことでどのようなことができるのか興味を持っている人	25 minutes	日本語 / Japanese	ハードウェア (Hardware)	Dai Miyamoto
Optimize your app for large screens	日本市場でも、タブレットや折りたたみ式デバイスがリリースされてきています。これらのデバイスでは、アプリはスマホと比べて大きなスペースを利用することができます。また複数のアプリを同時に使うマルチタスク機能も、スマホと比べて多く利用されることもわかってきています。 このセッションでは、タブレットや折りたたみ式デバイスのような大画面デバイスをサポートするにあたって考慮しなければならないこと、そしてサポートするための技術的な選択肢について紹介します。具体的には Jetpack Window Manager や、SlidingPaneLayout、Activity embedding のような大画面デバイスをサポートするために有用な API、レスポンシブレイアウトを実現するための方法、大画面デバイスをサポートする際に利用できるテスト手法について紹介します。	Android アプリ開発にある程度の経験をお持ちの方。Android アプリ開発の基本的な知識（例えば Activity のライフサイクルなど）を前提としたセッションです。アプリ開発の基本的な知識をお持ちでない方は、セッションを楽しんでいただけない可能性が高いように思います。 セッションで表示するサンプルコードを Kotlin で記述します。Kotlin の基本的な文法を知っている方であれば、よりセッションを楽しんでいただけると思います。	40 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Chiko
Gradle Managed Virtual Devicesで変化するエミュレータ活用術	皆さんはAndroidエミュレータを使っていますか？ 普段の開発であたり前のように使われているエミュレータですが、Gradle Managed Virtual Devicesの導入によって、Androidエミュレータを取り巻く事情が大きく変化しようとしています。 Gradle Managed Virtual Devicesは、Android Gradle Plugin (AGP) 7.2より導入された機能です。 この機能を使うと、build.gradleにテストを走らせたいAndroidエミュレータのスペックを書くだけでAVD (Android Virtual Device)の作成・起動からテストまでを1コマンド1つで実行できます。 また、(Canary版ではあるものの)Automated Test Device (ATD)と呼ばれる軽量なエミュレータも利用できるようになっています。 一見とても便利そうなGradle Managed Virtual Devicesですが、本機能にも得手不得手があります。 例えば、画面が表示されなかったり、テスト実行が終わると同時にエミュレータも終了したりするため、意図通り動かなかったときの解析が大変なことがあります。 本セッションでは、新しく導入されたGradle Managed Virtual Devices機能はどのようなものなのか、それによってエミュレータの使い方がどのように変化してくのか紹介します。 また、AGPの実装を調べて判明した内容を元に、Gradle Managed Virtual Devicesを活用する際に直面しがちなトラブルの解決方法も合わせて紹介します。 具体的には次のようなトピックなどについて紹介します。 これを機に、皆さんが持っているAndroidエミュレータの知識もアップデートしませんか？ ・Gradle Managed Virtual Devicesでできること、できないこと ・Gradle Managed Virtual Devicesの活用例 　・Robolectricでは心許無いテストを動かす 　・UIテストを動かす 　・スクリーンショットテストを動かす ・CI環境上でGradle Managed Virtual Devicesを使うときの工夫 ・Gradle Managed Virtual Devicesの実行で問題が起きたときの調査方法Tips ※できる/だけCanary版を含む最新のAGPの実装を調べた結果にもとづいて話す予定です。	・Gradle Managed Virtual Devices機能に興味はあるが、深く触っていない方 ・Gradle Managed Virtual Devices機能を試してみたものの、思った通りに動作せず諦めてしまった方 ・CI環境において、Android エミュレータ上でテストを走らせる方法を知りたい方	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	sumio_tym
Introduction to Wear OS Application Development	みなさま、ウェアラブルデバイスはお使いですか？ Google I/O 2022で、今秋にGoogle Pixel Watchが発売されることが公表されました。 Wear OSの開発をめぐる環境は、近年大幅にアップデートされています。 たとえばJetpack APIの登場や、Android Studio連携の強化、ドキュメントやコードラボの刷新などが行われました。 本セッションでは、Wear OSアプリのリリースを目標に、これを知っていればWear OSのアプリを実装できるという要点を紹介します。特に、Wear OSアプリの開発をこれから始められる初心者の方を想定し、Wear OSがどのようなもので、どんな機能が提供できるのかという点も詳しく解説します。 現在考えているセッションの内容は次のとおりです。 - Wear OSの開発について 　- Wear OSではどんなことができるのか 　- スマートフォンのアプリ開発との差異 　- Wear OSアプリのUIとUX - APIのサポート状況の概観 　- Jetpack API 　- Compose for Wear OS - Wear OSの開発の始め方 　- セットアップ 　- デバッグ方法 - Wear OSアプリをつくる 　- Overlays 　- Tiles 　- Complications 発表をとおして、最新のWear OSアプリの開発情報について共有できれば幸いです。	・Wear OS向けのアプリに入門されたい方 ・以前、Wear OS向けのアプリを開発されていた方 ・最近のWear OS向けのアプリの開発情報をキャッチアップされたい方	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Ryo Yamazaki
Androidのモダンな技術選択にあわせて自動テストもアップデートしよう	ここ数年の間に、Jetpack ComposeやKotlin CoroutineおよびKotlin Flow等、Androidアプリ開発をする際に選択できる新しい技術が登場しました。現在では、多くのプロダクトがそれらの技術の導入や移行を進めています。 新しい技術を選択する際には当然キャッチアップが必要になりますが、あわせてその技術を使ったコードをどのようにテストするかを知ることも重要です。テストする手段を知ることで、新しい技術を使ったコードの動作確認も容易になりますし、その技術を更に理解することにも繋がります。とはいえ、プロダクトコードへの導入を進めながら、テストの書き方までキャッチアップするのが大変なことも事実です。 本セッションでは、新しく採用されるケースが多い、次の技術のテスト方法を紹介します。また、より実践的な内容にするために、テスト方法をGuide to app architectureで推奨されるレイヤー毎(UI Layer・Data Layer)に整理して紹介していきます。 ・Kotlin Coroutine ・Kotlin Flow ・Jetpack Compose ・Dagger Hilt 扱うトピックが幅広いため、1つ1つのテスト方法について深堀りするよりも、それぞれの技術・レイヤーの最初のテストを書くために必要な知識を整理し、テストを書き始めるハードルを下げることを目指します。	・Kotlin CoroutineおよびKotlin Flow/Jetpack Compose/Dagger Hiltの導入をしたが、自動テストのキャッチアップができていない方 ・Kotlin CoroutineおよびKotlin Flow/Jetpack Compose/Dagger Hiltを導入予定で、自動テストの基本的な書き方に興味がある方	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Nozomi Takuma
BLEを使ったアプリを継続的に開発するために	BLE、Bluetooth Low Energyとは近距離無線通信規格Bluetoothのうち低消費電力で利用できる通信モードで、IoTデバイスとスマートフォン間の通信などで広く利用されています。 AndroidというOSにおいては、端末依存の挙動やコールバック地獄などが存在するおちゃめなAPIとしても一部で名が通っています。 IoTデバイスを開発している企業にAndroidエンジニアとして入社した場合、BLE APIと付き合っていくことになります。現在提供されているコールバックベースのBLE APIを受け入れて真摯に向き合っていくのはおすすめできません。RxJava時代を経て今我々の手にはCoroutineという素晴らしい道具があります。このセッションではBLE APIを利用したデバイスとの通信の実装を解説した後に、私が経験したCoroutineを利用したメンテナンス性の高いコードへの書き換え事例やプラクティスを紹介します。 # こんな人に聴いて欲しい - BLE APIを利用するアプリを開発していて、コードのメンテナンスに疲れてしまった方 - これからBLE APIを利用するアプリを書こうとしている方 - メンテナンス性の高いBLE APIを利用するコードを書きたい方 # 話すこと - BLEに対応したデバイスとの通信を実装する方法 - 2022年、Coroutine/Flow時代のBLE APIプラクティス # 話さないこと - プロトコルの詳細 - BLE APIのおちゃめさ加減 # こんな人を待ってます - BleXを開発してくれるGoogler	BLE APIを利用するアプリを開発していて、コードのメンテナンスに疲れてしまった方 これからBLE APIを利用するアプリを書こうとしている方 メンテナンス性の高いBLE APIを利用するコードを書きたい方	25 minutes	日本語 / Japanese	Android Framework	Moyuru

今年リリースされたCTS-D (互換性テストスイート)をアプリ開発に活用しよう！	Androidアプリを開発していて、あるデバイスでは動作するのに、他のデバイスでは、予想通りに動作せず、苦労したという経験はありませんか？ このセッションでは、デバイスの互換性をテストするCTS-Dについてご紹介します。CTS-Dは、グーグルが今年6月にリリースしたプログラムで、アプリ開発者の方に、互換テストの共同開発に参加していただき、互換性の問題を、Androidコミュニティーとして、一緒に解決していくことを目指して発足されました。	Androidの開発者全般。初心者～上級者。 CTS-D（互換性テストスイート）の基本から、テスト開発への参加方法までを網羅します。	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Sachiyo Sugimoto, Saryong Kang
State of Jetpack Compose	Hear about Jetpack Compose from the Compose team. What's new in Compose 1.2, what's coming in 1.3 and tips you can use to adopt Compose in your app.	All levels of Android developer	40 minutes	English	Jetpack Compose	Ben Trengrove
Lessons Learned Migrating the Maps SDK to Compose	This session goes over the story behind Maps Compose—the Jetpack Compose interoperability library for the Maps SDK for Android. Learn lessons in Compose API design and how to effectively migrate your library or app to Compose. At the end of this talk, developers will know how to: * use interoperability APIs to bridge View-based apps/libraries to Compose * follow best practices for composable elements' API design * use subcomposition to tap into Compose APIs for managing tree-like data structures	Intermediate developers. Have prior knowledge of Jetpack Compose.	40 minutes	English	Jetpack Compose	Chris Arriola
エンジニア駆動でデザインツールの刷新ができた	昨今では、アジャイル開発でのスタイルが主流となりつつある中で、デザイナーとエンジニアは密なコミュニケーションを取りながら要件決めやプロダクトの実現性や設計などを一緒に作っていく必要があると思います。 デザイナーとのキャッチボールがうまくいかないと開発スピードが十分に出ない事や要件の認識違いなど致命的な問題に遭遇することもあります。 そういった問題をツールの刷新で解消できたお話をできたらと思います。	エンジニア・デザイナーとのコミュニケーションやインテグレーションで困っている方 ツールなどの刷新をするきっかけに困っている方	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	bami
Android アプリの内と外をつなぐ UI	UI はアプリの中だけのものではありません。ホーム画面からアプリを起動する際のスプラッシュ スクリーン、ステータス バーやナビゲーション バーとアプリ内 UI の干渉を防ぐ WindowInsets など、アプリの UI がシステムの UI と連携する場面は数多くあります。最新の Android 13 でもバック ジェスチャーに関する API が変更され、対応が必要になっています。このセッションでは、アプリの内と外をつなぐ UI に焦点を当て、最新のベスト プラクティスを解説します。	View また Compose でアプリを開発している人	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	荒木佑一
Android Vitalsのデータを自動監視してビジネス指標を向上させよう	Android Vitalsを使用すると、アプリの安定性やパフォーマンスに関する指標を確認できます。 Vitalsの指標の低下は、ユーザ体験の低下に繋がるだけでなく、Playストアでの評価の低下や検索ランキングの低下にも繋がります。 https://developer.android.com/distribute/best-practices/develop/android-vitals?hl=ja GoogleI/O 2022ではVitalsの指標であるANRを50%減らすと、20%のリテンション率向上、30%トランザクション数が向上したプロダクトが紹介されていました。 https://www.youtube.com/watch?v=DYdHLqLVspY したがって、Vitalsの指標はUUやCVRなどのビジネス指標にも影響する可能性が高いため、指標の改善・維持はプロダクトにとって重要な役割となります。 しかしながら、Vitalsの指標を注意深く監視できていなかったり、監視の自動化が難しいため手動での監視となってしまっているのが実情ではないでしょうか？ 2022年3月にPlay Developer Reporting APIが公開され、Vitalsの指標をAPI経由で取得することが可能になり、監視の自動化や分析をより簡単に行うことが可能になりました。 https://android-developers.googleblog.com/2022/03/play-developer-reporting-API.html 本セッションでは、監視の自動化方法のみならず、アラート発生時の弊社の運用方法ついても紹介する予定です。 <アジェンダ(予定)> ・Android Vitalsについて 　・各指標について 　・(Crashlyticsとの使い分けについて) ・Vitals指標を改善するメリットについて ・Play Developer Reporting APIについて ・監視の自動化方法について ・アラートが発生した際の運用方針について ・(よく見かけるクラッシュとその対処方法の紹介) ・導入の効果・知見	・Android Vitalsについて知りたい方 ・Vitals指標の重要性について知りたい方 ・Play Developer Reporting APIについて知りたい方 ・弊社でのVitals監視の自動化方法と運用について知りたい方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Yusuke Suzuki
Kotlinize your CI, from Gradle tuning to Kotlin Script	This talk will explain how I leveraged Kotlin in various ways to write as much automation code as possible with this language, and so being easy to maintain for any Kotlin user. This will introduce several features of Kotlin (like Kotlin script) or Gradle (like Composite Builds, the JavaExec task, Gradle Plugins, etc)	People interested to know more on the CI side of their project (Android or not), or already doing it but facing issues with maintainability due to the various languages used in 1 project (Bash/Python/Ruby/etc).	40 minutes	English	保守・運用・テスト (Maintenance, Operations, and Testing)	Pierrick Greze
Compose for Desktopで始めるAndroid開発効率化ツールの作成	Compose for DesktopはJetbrainsが開発するデスクトップアプリ向け（macOS、Linux、Windows）のUIフレームワークです。Compose for DesktopはJetpack Composeの仕組みを土台にしており、Jetpack Composeと同じ仕組みでデスクトップアプリを作成できます。そのためJetpack Composeを習熟したエンジニアであればCompose for Desktopにて簡単にデスクトップアプリを作成できるようになっています。 というようにCompose for DesktopはAndroidエンジニアにとって学習コストが低く扱いやすいデスクトップアプリ向けのUIフレームワークですがAndroidアプリ開発における使い所がないなどいう認識の方も多いのかなと思っております。なのでこのセッションではCompose for DesktopとADBを組み合わせたツール開発の事例を取り上げ、Androidアプリ開発でどのように活用できるか解説をしたいと思います。また今後利用を考えている方のためにCompose for Desktopを利用した開発の良さや辛さなどもあわせて取り上げようと思います。	Compose for Desktopを利用したツールの作り方を知りたい方	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	Yusuke Katsuragawa
実践 Glass EE2 向けアプリ開発	Google Glass のエンタープライズ向け最新モデル「Glass EE2」の販売が開始されて約3年が経過しました。 グローバルではいくつかの採用事例がありますが、国内ではまだマイナーな Android デバイスです。 一方で、このようなスマートグラスに対するユーザーの期待値は高く、近年は様々なメーカーからデバイスが販売されています。 本セッションでは、スマートグラスに期待されているユーザー体験を解説した後、Glass EE2 向けの開発環境やデザインガイドライン、アプリ開発例を紹介します。	・ニッチな Android デバイスが好きな方 ・スマートグラスに興味がある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	shanonim
長期運用アプリのリファクタリングを考える	Android始まって以来の大きな転換期を迎えている今、リリースしてから年月が経過している長期運用アプリの中にはそろそろ構造的に無理が生じているものも増えてきているのではないでしょうか？ Jetpack Composeが発表され、アプリアーキテクチャガイドでベストプラクティスが案内されていますが、試行錯誤を繰り返しながら運用しているアプリではなかなかついていくことに苦労することも多いです。運用中のアプリの開発を止めるわけにはいかずフルスクラッチという選択肢が取れないこともあるでしょう。 このセッションでは - ゴッドアクティビティが乱立しているアプリの分解方法 - 段階的なJetpack Compose化 - リファクタリングにおける効率的なAndroid Studioの使い方 など、アプリのアーキテクチャの方向性が明確化してきた中で長期運用しているアプリをリフレッシュする方法を考えます。	長期運用しているアプリを抱えているエンジニア	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	shinmiy
社内でのモバイルアクセシビリティ推進	担当アプリのアクセシビリティが気になっていても、 会社と言う環境で作るアプリはいろいろな人が関わっているため、新機能と比べてあまり魅力的でないものを簡単に取り込めないですね。 それにGoogleのガイドラインやコードでの対応方法を知っていても、Androidエンジニアの自分だけで出来る事の限界があります。 本当はデザイナー、PMやQAなど、アプリ関係者のみんなを巻き込む必要がありますのに、 実際いざとなると、どこからどうすれば良いか分からないし。。 Webアクセシビリティのガイドラインを読んでもピンと来ないし。。 デザインとかiOSなど自分の担当領域以外のことは詳しくなくて質問に答えれないし。。 心が折れてしまいそうですね。 せっかくアクセシビリティ強化を考えている方がそんな壁にぶつからないように、本セッションでは、現在社内でアクセシビリティを推進している私の通ってきた道と、集めてきた情報を話します。 絶対に聞かれる「なぜアクセシビリティ対応をするか」の答え方、社内勉強会の内容、対応範囲の決め方や具体的な取り込み方など、必要な参考情報を共有します。 ご自身の担当アプリのアクセシビリティ改善がうまく行きますように！	アクセシビリティにちょっと興味があるけど、仕事にどう取り込めるか分からない\Androidエンジニアの方。 担当アプリのアクセシビリティを改善するために、推進方法の参考が欲しい\Androidエンジニアの方。	40 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Tiphaine (ティフェン)
All your Compose @Previews to screenshot tests without instrumentation	Screenshot tests are great way to test your view layer and catch many regressions. Jetpack Compose already has a handy @Preview feature for the IDE - and you can automatically make then into screenshot tests. With the help of Paparazzi library, you can capture screenshots without device or emulator, which is cumbersome on CI. During this session I will share our screenshot testing experience from a large enterprise app including GitHub Actions CI integration.	The talk is accessible to everyone, but interest in Android testing is welcomed.	25 minutes	English	保守・運用・テスト (Maintenance, Operations, and Testing)	David Vávra
マンガアプリのメモリ改善とメモリ解析方法	1. Lineマンガで8年運用したAndroidアプリのレガシーの問題としてOOMのメモリクラッシュ問題を真剣に向き合って解決した事例を紹介します。 2. AndroidのVMの発展過程の紹介とそのメモリ管理方法とGCの管理方法の変化を辿る。3.基本的なGC(Garbage collection)の考え方からAndroidのメモリ解析のための基礎知識を伝える。4.具体的なメモリ解析の方法を事例で紹介する。5. メモリリークを防ぐため、メモリを改善するための注意点と実際の例を紹介する。6.Google I/O 2022のLarge Screen意識でさらに重要になったConfiguration changeのメモリ対策	JVMメモリ管理方法、GC(Garbage collection)に関心がある方、 Android,メモリ、パフォーマンスを改善したい方、 マンガが好きな方。	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	sai.choko
AndroidManifest.xmlはその後どうなっているのか？	Android開発している必ず編集するであろうAndroidManifest.xmlですが、このxmlファイルがその後どうなっているかご存知でしょうか？ アプリに組み込まれて必要に応じて適宜参照されるのかなないライメージはあるかもしれませんが、実際に追ってみたことはありますか？ このセッションでは、AndroidManifest.xmlがどうアプリに組み込まれ、設定した値がどのように使用されているのかを解説します。	- Android開発をしている人 - AndroidManifest.xmlがどう使われているか知らない人	25 minutes	日本語 / Japanese	Android Framework	takasy

Android(AOSP)でウェアラブルIoTデバイスを作ってみた	私たちはウェアラブルIoTデバイスを開発するにあたり、OSとして Android Open Source Project(AOSP)を採用しました。 本発表ではAOSPを採用するに至った経緯や、ウェアラブルIoTデバイス として開発する上で苦労した点・工夫した点について紹介させていただき ます。 具体的には以下のような内容について紹介します。 - デバイス内のアプリをマイクロサービス的に設計・実現 - Play Store・Google 開発者サービスを使わずに運用するためのアレコレ - 画面がない！困ったことと対処 本デバイスは現時点で数千台が世に出ており、特に画面がないことにつ いては現在もデバイスを運用しながら良い方法を模索し続けています。 デバイスを実サービスとして運用する上での試行錯誤についてもご紹介 させていただきます。	- 組み込み（embed）Android技術者 - 独自デバイス上でAndroid(AOSP)を動かすことに興味のある方 - IoTデバイスを使ったサービス開発・運用に興味がある方	25 minutes	日本語 / Japanese	Android Framework	Yoshinori Mukai
5chマイクを使ってビームフォーミングできるようにAudio Frameworkを 魔改造する	ビームフォーミングとは複数のマイクを使って、特定方向の音声を強調し つつそれ以外の音声を抑制した音を取り出すことを実現する技術であ り、モバイルPCやWebカメラ、ヘッドセットなどにも使われている技術で す。 ビームフォーミング処理は、専用ハードウェアチップで実装されている ケースが一般的ですが、ソフトウェアで実装することも可能です。 私たちは、Androidベースのデバイスでもビームフォーミングを扱えるよう にするべく、5chマイク（5つのマイク）を扱えるようにしつつビームフォーミ ングライブラリを組み込むために、Android OS のAudio Framework と 格闘したのでその記録を紹介します。 本発表では、ビームフォーミングの仕組みやAudio Frameworkの構成な どにも触れつつ、実際にどのような作業によってビームフォーミングを実 現したのかを説明します。 具体的には以下のような作業を行う必要があります。 - Audio Framework にビームフォーミングのライブラリを組み込む。 - 5chマイクからの入力を扱えるようにする - アプリからは1chのデータのみが入力されているように見えるようにす る - 5chマイク以外からの入力（イヤホンマイクやBluetoohヘッドセット）との 切り替えをいい感じに実装する 音声処理に興味がある方だけでなく、普段あまり触れることの少ない Android Frameworkを直接改造するという試みに関心のある方にも興 味深い内容になると思います。	- 組み込み（embed）Android技術者 - 独自デバイス上でAndroid(AOSP)を動かすことに興味のある方 - 特にマイク・音声処理周りに興味のある・仕事にしている方	25 minutes	日本語 / Japanese	Android Framework	Akira Kimura
実例から学ぶJetpack Composeのパフォーマンス改善	Jetpack ComposeはAndroid Viewシステムの教訓を経て、多くのケー スで優れたパフォーマンスを発揮します。一方で、いつの間にかパフォー マンスが低下している経験をした人も多いのではないでしょうか？特にア ニメーションの表現や、スクロールやドラッグの操作によってUIを切り替 える際により問題が発生します。これらは、必要以上の広範囲で高速に 更新処理が行われることに起因します。他にも、LazyリストやViewPager、 RecyclerViewとJetpack Composeを組み合わせる際等にも注意すべき 点があります。 より良いパフォーマンスを得るには、Jetpack ComposeのUI更新の仕組 みを理解し、状況に応じて対処していく必要があります。このセッションで は実例をもとに、パフォーマンス低下の原因、計測方法、改善方法につ いて解説を行います。	・Jetpack Composeを既に利用している人 ・Jetpack Composeにおいてパフォーマンスに困っている人	25 minutes	日本語 / Japanese	Jetpack Compose	Mori Atsushi
2022年の動画再生アプリの作り方	2021年10月にJetpack Media3が発表されました。 これまでは別々のライブラリとして提供されていたメディア関連のAPIが 単一のライブラリに統合され、動画再生に関する様々なユースケースを 容易に実現できるようになりました。 また、近年ではピクチャー・イン・ピクチャーへの対応や折りたたみ式端 末への対応など、動画再生アプリに対しての機能要件が多様化してきて います。 本セッションでは、2022年に動画再生アプリを作る際に考慮すべきこと や具体的な実装方法について紹介します。 予定している内容 ・Jetpack Media3とExoPlayerについて ・MediaSessionServiceを用いたバックグラウンド再生 ・ピクチャー・イン・ピクチャーへの対応 ・折りたたみ式端末への対応	・Androidのメディア系APIに興味がある方 ・動画再生アプリの品質向上に興味がある方	25 minutes	日本語 / Japanese	Jetpack	oidy

通知チャンネルに対応する	通知はアプリの外でユーザーとコミュニケーションができる重要な機能で
	す。しかし、ユーザーに通知を受け取ってもらうハードルは年々高くなっ
	ています。
	通知機能にはさまざまなアップデートが行われており、Android 8からOS
	で通知チャンネルを作成できる機能が追加されました。細かく通知を管理
	する仕組みが提供されたことで、ユーザーが通知の許諾のコントロー
	ルをアプリの内外でしやすくなりました。またAndroid 13では通知の
	Permissionが追加され、ユーザーが必要とする通知を選択できるよう
	になりました。
	
	本セッションでは実際に弊社のアプリで行ったチャンネル対応の事例を
	通して通知設定の課題・解決方法を共有します。具体的には次のような
	内容を想定しています。
	- 通知チャンネルとは
	  - なぜ通知チャンネルに対応するのか
	- 通知許諾とは
	  - Android13と12以下の挙動
	- チャンネル対応
	  - 既存のアプリ内画面との両立
	  - マイグレーションに当たっての課題
	  - アプリ内の通知画面
	    - 表現方法
	    - ユーザー毎の設定
	  - 実装上の課題
	    - 連打制御
	- 効果検証
	  - 許諾の変化
	
	魅力的なサービスの情報をお届けするため、よりユーザーが使いやすい
	通知の設定を実現していきましょう。
		通知チャンネル対応を検討している方
		通知許諾を高めたい方
			25 minutes	日本語 / Japanese	Android Framework	Miyabi Gouji

Android "を" ビルドしてAndroid Systemを覗いてみよう	Android開発者の皆さんは日々Android アプリ (apk / aab) をビルドして いますが、Android"を"ビルドしたことがありますか？Andriodをベースと する組み込みデバイス等の開発に従事している方は、日常的にビルドし ているかもしれません。 Androidはオープンソースでコードの大部分が公開されており、Android そのものをビルドし、システムイメージを作ることが可能になっています。 また、カスタムのカーネルをビルドし、それを前述のシステムイメージに埋 め込むことも可能です。 システムイメージ・カーネルをビルドすること自体は普段のサービスアプ リ開発において、役に立つ機会は少ないです。しかし、ただ何となくみて いたAndroidシステムの挙動の裏側を鮮明に理解することができ、今後 のアプリ開発に役に立つ機会がきっと来るでしょう。 本セッションでは、サービスアプリ開発者であるスピーカーの視点から主 にアプリ開発者に向けて、Androidのシステムイメージのビルド方法、 カーネルのビルド方法を Android Developers に記載されている行間を 補完しながら step by stepで進めていきます。 その後、コードの一部を修正しながらAndroidシステムの動きに迫ってい きます。 <予定しているアジェンダ> - Android システムについての簡単な概要 - Android のビルド方法 - カーネルのビルド方法 - 実例1. Android システムを修正しながら確認する navigation bar の振 る舞い - 実例2 :Android システムを修正しながら確認する Dialog の振る舞い	Androidのシステムそのものに興味がある人 Androidをビルドしてみたい人 Androidのビルドに挫折した過去がある人	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Kazuki Chigita
アプリエンジニアとQAチームがデバッグ機能の改善に取り組むぞ！	アプリのデバッグ機能を作る際に、エンジニアがデバッグのために作る場合と、 QAやPdMチームなどの要請によって、作る場合があると思います。 サービスの運用が長期化されて、デバッグ機能が乱立し、 何がどう動いているデバッグ機能なのかのドキュメントの整理もままならな い状態から、 QAチームが今必要としているデバッグ機能をリサーチ、 QA工数削減した、GOユーザーアプリ改善チームの取り組みを紹介しま す。 予定しているセッション内容 - GOユーザーアプリチームの開発体制 - GOユーザーアプリに求められたデバッグ機能 - 乱立したデバッグ機能を整理して、QAチームを今必要なデバッグ機能 について考える	- QAチームがあるサービスを開発するエンジニア - デバッグ機能を実装している、実装したいエンジニア	25 minutes	日本語 / Japanese	開発体制 (Development Process)	Satoru Komai
移りゆくデファクトスタンダードにチームとしてどう追従するか	モダンと呼ばれるような技術・設計は開発における生産性や諸リスクに 有効であることは周知の事実です。 一方でそれら理想を導入するための障壁を感じてしまい、プロダクトコー ドが未だレガシー環境であることに悩まされている方も少なくないと思い ます。多くの場合は、モダンとレガシーとのギャップが大きいが故に導入 に至るまでのロードマップ策定が行えない場合が多いと感じます。 MVVMをはじめとしてマルチモジュール化からJetpackCompose導入ま で、導入を目指すには少なくない考慮が必要ですが適切なステップを踏 むことによって、着実にそして確実にモダン環境への移行を進めていくこ とが可能です。 レガシーからモダンへと移り変わるための各種ガイドラインの策定とロー ドマップの作り方について、個人としてではなく「Androidチーム」として進 められるよう、知見を実体験とともに解説したいと思います。	Androidチームとして技術的負債の返却、レガシーからの脱却を考えて いる方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	tk-masuda

はじめようビルドメトリクス	オレはいったいどれほどの時間をこのプログレスバーと共に過ごしてきた
	んだろう…
	
	Android開発とビルドは切っても切り離せない関係です。
	このセッションでは手元のマシンから収集できる情報を集めて、どれほど
	の時間を費やしているのか可視化する方法を指南します。
	
	- 1回の分析
	  - 「ビルド」定義の確認
	  - gradle-profilerを使ってみよう
	- 1日の分析
	  - 簡単なAPIを作成してのビルド回数などの収集方法
	
	このセッションでは「日常のビルドにかかる時間を知る」ことにフォーカス
	しているため、分析結果を使った高速化の方法などには触れません。
		- 新しいPCを買ってもらおうとしたら上司に「本当にそんなスペック必要な
		の？」と言われてしまった人
		- チームのスループットをあげるためになんでもしたい人
		- ただ己の時間が何にどれほど費やされているのか知りたい人
		- 「gradle-profilerのことは知らないけどなんか面白そうだな」ぐらいの人
		
		Androidアプリをビルドしたことがあれば前提知識は不要ですが、gradle-
		profilerの深い話や高速化のノウハウなどはありません。
			25 minutes	日本語 / Japanese	開発体制 (Development Process)	こまたつ

CameraX × ML Kit でパスポートOCR機能を実装	海外旅行を予約する上でパスポートは必須の情報ですが、自身および同行者の名前や生年月日やパスポート番号などの情報を入力することは、ユーザーにとって少なからずの手間となります。 弊社が開発する海外旅行の予約アプリでは、その手間を軽減すべく、パスポートOCR機能を開発しています。 実装にはJetpack Composeにも組み込みやすいCameraXとML Kitの組み合わせを採用しています。 CameraXはカメラ機能を手軽に開発できるようにしたJetpackのライブラリで、ライフサイクルにバインドして動作します。 ML Kitは機械学習を利用した開発を手軽におこなえるようにしたSDKで、以前はクラウドで動作するAPIと共にFirebaseで提供されていましたが、オンデバイスで動作するライブラリはML Kitとして独立しています。 弊社アプリのパスポートOCR機能では、CameraXのプレビューと画像解析のユースケースと、ML KitのText Recognition（文字列検出）を利用しています。 それらライブラリを用いてパスポート情報を取得する実装の内容や、精度向上のためにおこなっている取り組み等を紹介します。	- CameraXに興味がある方 - ML Kitに興味がある方 - UXの改善に取り組んでいる方	25 minutes	日本語 / Japanese	Jetpack	Yuto Akaike
2022年のBytecode Weaving ~Google、Transform API やめるってよ~	いくつかのライブラリでは、Androidアプリケーションのビルド時にBytecodeを書き換えることにより魔法のような機能を提供しています。このBytecode書き換えをBytecode Weavingと言いますが、Bytecode Weavingを実現する際にこれまで使われてきたTransform APIが近い将来リリース予定のAndroid Gradle Plugin 8.0で削除されることが予告されました。 本セッションではBytecode Weavingで何が実現できるのか、またTransform APIの何が問題だったのかを解説します。その後、Transform APIに代わって導入されるInstrumentation APIがどのように問題を解決しているかを具体的な使い方とともに解説していきます。 Bytecode Weavingはライブラリ提供者にとってとても強力な手段です。何ができるかを知っていることでライブラリが実現できることが格段に増えるので、ぜひこのセッションで最新の使い方を学び便利なライブラリを公開していただけたらと思います。	ある程度build.gradleはいじったことがあり、魔法のような機能を提供しているライブラリが裏でどのようなことをおこなっているかに興味がある方、またそのようなライブラリを作ってみたいと思っている方を対象としています。	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	zaki50
プロダクトで安全にDataStore移行する	Jetpackライブラリの DataStoreは、アプリ内のデータストレージのライブラリです。SharedPreferencesと同様にkey-valueでデータを格納することが可能ですが、SharedPreferencesより優れた点があり、DataStoreへの移行が勧められています。 しかし実際に移行を考えると、なかなかモチベーションが湧いてこないかも知れません。 例えば、以下のようなことでモチベーションが低下しているかも知れません。 ・SharedPreferencesの使っているkeyが多すぎる ・移行しなくても不便がない また、マイグレーションガイドもあり、移行は容易そうですが、実際にプロダクトで移行を進めるためにはいくつか懸念があります。 ・SharedPreferencesでたくさんのkeyを用いているが、全てマイグレーションが可能なのか ・一部のkeyだけマイグレーションして、段階的に移行が可能なのか ・DataStore移行後にrevertした時にデータの不整合が起こらないか そこで、本セッションでは、DataStoreへの移行するモチベーション、移行方法、実際のプロダクトで移行した際に起こり得る懸念とその対応について紹介をします。 概要 ・DataStoreの概要 ・DataStoreに移行するモチベーション ・SharedPreferencesからのマイグレーションについて ・実際のプロダクトでの移行計画 　・段階的な移行 　・テスト・QC（QA） 　・DataStore移行後に起こりうる懸念 ・その他（可能であれば） 　・Preference DataStoreで保存したものをProto DataStoreでの保存に移行できるのか 　・SharedPreferenceのkeyの管理について	SharedPreferencesを使用したことがある方。 DataStore移行を検討している方。	25 minutes	日本語 / Japanese	Jetpack	Go Takahana
2022年ヘルスケアアプリのつくり方	健康管理を目的とした様々なヘルスケアアプリが開発されています。 ヘルスケアアプリでは端末のセンサーやカメラ、ウェアラブル機器などを使用して心拍数、運動時間、睡眠時間、摂取カロリーなど様々な種類のデータが扱われています。 このような健康状態や運動に関するデータは様々なプラットフォームで管理されており、 サービス自身が保持していないデータについてもプラットフォームを介して参照することで、 健康管理の体験を提供できるようになっています。 その一方で、Androidでは様々なヘルスケア管理プラットフォームが存在し、使っているサービスによっては異なるプラットフォームに計測データが保存される場合もあります。 また今年のGoogle I/Oでは新しいプラットフォームであるHealth Connectが発表され、複数の管理プラットフォームとのデータ連携ができるエコシステムとして期待されています。 本セッションでは、2022年に新しくヘルスケア アプリを開発した際に得られた知見をもとに、 ヘルスケア プラットフォームHealth Connectとの連携アプローチについて話せればと思います。 話すこと ・Google FitとHealth Connectの機能比較 ・iOSのHealth Kitとのデータ互換性を考える ・視点点で連携すべきプラットフォームとして検討した点	ヘルスケアに関わるアプリ開発に興味のある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Kohei Yamamoto
Jetpack Composeの状態管理とAPIコール - React Hooksに習う、Apollo + Jetpack Compose	Jetpack Composeは、思想的にReactに近いところがあります。Composable関数とReactの関数コンポーネント、hooksはその最たる例と言えるでしょう。 また、GraphQLクライアントのApolloは、GraphQLの通信のレスポンスをグラフ構造そのままにキャッシュとその更新を可能にしています。Apolloのキャッシュ機構は、単なるGraphQL APIのコールのみならず、状態管理ライブラリとしての役割も果たします。これにより、フロントエンド開発者はAPIコールやキャッシュについて複雑に考えることが減りました。 しかし、Apollo KotlinにはApollo JSで用意されているcustom hooksがないなどの差異により、React + Apolloのような開発者体験がJetpack Compose + Apolloの組み合わせでは得られません。 このセッションでは、React Hooksを参考にApollo Kotlinをラップし、シンプルにAPIコールとキャッシュ、状態管理を行うアーキテクチャを実現する方法について発表します。 - Jetpack ComposeとReactの類似性について - GraphQLとApolloについて - Apollo JSとApollo Kotlinの差分について - Apollo KotlinをJetpack Composeに適した形でラップする - ApolloとJetpack Composeで実現する状態管理アーキテクチャについて	GraphQLに興味のある人 Apolloに興味のある人 Jetpack Composeにおける状態管理に興味のある人	40 minutes	日本語 / Japanese	Jetpack Compose	mayamito
Deep dive into Jetpack Compose Text	我々Android Textチームは新しいUI TookitであるJetpack Composeのテキスト描画APIを設計し、実装しています。テキストは多くの部分をプラットフォームのAPIに頼っており、どこまでがライブラリ内の実装で、どこからがプラットフォームAPIを利用しているのかがわかりにくくなっています。この発表では最新のテキストに関連したJetpack Composeの機能を、APIとその内部構造を中心に解説していきます。 Our team has been developing a new UI toolkit called Jetpack Compose. The Android Text team has been designing and implementing the Text rendering APIs in Jetpack Compose. Text components are still relying on lots of platform APIs, so it is quite hard to find a line between library implementation and platform APIs. In this session, I'm going to introduce the new feature in Jetpack Compose and what happens under the hood.	Jetpack Composeの内部構造に興味がある方 Androidがどうやって文字列を描画しているのか興味のある方 People interested in the internal details of Jetpack Compose. People interested in how Android draws text on the screen.	25 minutes	日本語 / Japanese	Jetpack Compose	Seigo Nonaka
Considerate App Update Delivery	リリースフローを自動化したことで安全なリリースだと思いこんではいませんか? 自動化によって属人化やヒューマンエラーが減った冪等性のあるフローなら、確かにそれが動いている間は安全と言えるでしょう。 しかし障害等で手動でのリリースが必要になったとき、その自動化されたフローから復元した手動リリースは安全ですか? 安全なリリースフローとはヒューマンミスの排除に限りません。例えば required feature や dangrous permission の追加はアップデート配布対象が減るだけでなく、自動アップデートをオフにしてしまいます。それが意図的であったとしても、リリース前に妥当性及び副作用を確認出来ることが望ましいでしょう。 多くのリリースフローは最小に近い手順を自動化しているだけで、上記のようなケース・検証の穴が十分にカバーされていないように思います。それらの検証が欠けているとき、そのリリースフローは安全とは言い切れないかもしれません。あるいはリリースよりもずっと前から継続的に行う検証が必要かもしれません。 本発表ではAPK/AAB から静的に取得出来る情報、端末で取得出来る情報、そしてファイルからはどうしても取得出来ない情報を理解し、それぞれの情報を安全なリリースの実現やリリースフロー自体の改善に利用する試みを紹介します。 - APK/AAB から静的に取れる情報 - ランタイムで取れる情報 　- FireOS などの例外 - アプリファイルには含まれない情報 - 情報の利用方法 　- file metrics 　- ライブラリアップデート時の proguard/R8 rules のレビュー ※ いわゆるアプリの品質、QAについては言及しません。	安全なリリースに興味があるひと proguard/R8 rule の変更について、実際に動かすことだけが唯一の検証だと思っているひと 正直手動でリリースするとの方法を忘れてしまったひと	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Jumpei Matsuda
人の声を可視化する	ひとの声を可視化するとき、波形型で表示するのが一般的です。 Android端末のマイク性能にばらつきがあることによって録音時の波形が端末によっては小さく、音声が入っている箇所が分かりづらかったり、一定数秒間に表示する波の数がずれることがあります。 人の声を可視化するオーディオビジュアライズについて、Androidで起こりやすい課題と対策を紹介し、またAudio処理に関わるアーキテクチャ、AudioRecordの取り回し方の注意点などについてもご紹介するつもりです。	こんな人におすすめ ・これからAudio処理を学ぶAndroidエンジニア ・人の声を取り扱うプロダクトに参画するAndroidエンジニア 受講者が得られる知見 ・Audio機能の基礎おさらい ・Audio処理を実装する際の注意すべき点と最適化手法 ・Audio処理まわりで端末依存における条件分岐を実装したことがある方	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Miyuki Onuma

ついにアプリ毎の言語設定がOSレベルでサポートされたぞ！	アプリの多言語化対応を実施したことはありますでしょうか？ OS の言語設定に従って切り替えるだけで良いのであれば、各言語毎のリソースを用意し、適切にフォルダ分けをするだけで対応が完了します。しかしながら「アプリ内で言語切り替えできるようにしてよ」と要求されることがままあります。 これまでアプリ内言語切り替え機能を提供するには OS バージョン毎に異なる複数のワークアラウンドを用意する必要があり、大変な苦労を強いられてきました。 しかし Android13 および AppCompat 1.6.0 においてようやく OS レベルでアプリ毎の言語設定がサポートされることになりました。 本セッションでは、過去の OS における独自の言語切り替え機能を提供する辛さを解説し、その上で今回提供されたアプリ毎の言語設定が素晴らしいかを褒め称えつつ、多言語化を実施する際の注意点を解説します。	アプリの多言語化を考えている方 独自の言語切り替え機能で消耗している方	25 minutes	日本語 / Japanese	Android Framework	sobachanko
モバイルアプリのユーザープライバシー新時代	近年、アプリ開発においてもユーザープライバシーの重要性が増してきています。セッションではモバイルアプリでのユーザープライバシーへの配慮方法を紹介し、実装の指針を紹介します。 基礎的な知識から実装方法、ユーザーフレンドリーな仕様をカバー対象とし、将来導入されるPrivacy Sandboxといった新しいアプローチへの理解を推進することが目標です。 - パーミッションの制限、ロケーションの適切な取り扱い方法 - センシティブデータの管理・推奨手法 - デバイス識別子や行動分析などでのユーザートラッキングの注意点 - プライバシーに関するリスクを最小限するプラクティス たとえばパーミッションでは2022年1-3月にかけて「使用していないアプリの権限を削除」する挙動に変わっています。 これはユーザーのプライバシーを保護するための変更ですが、日頃からアップデートを追いかけていないと気づくのは難しいでしょう。 さらにプライバシーに対する理解度は組織、チーム、個人でさまざまなのが現状です。 プライバシーを尊重した機能、その導入の背景、ユーザーにとって嬉しい仕様を理解してチームで議論をリードできると強みになるはずです。この機会に知識を更新しましょう！	- プライバシーに関心がある方 - プライバシーは大事だと感じつつも、詳細をまだ調べきれていない方 - ユーザー満足度の高いアプリにしたい方 - Google Play Storeから不意にプライバシー違反で怒られたくない方	25 minutes	日本語 / Japanese	Security / Identity / Privacy	mhidaka
コルーチンを使って処理の見通しをよくする - リアルタイム放送の品質を向上し、保つために	私が現在開発している「リアルタイム音声放送」では、アプリとサーバー間の通信だけでなく、同時にアプリ同士のP2P通信も行っており、複数の通信状態を非同期に、かつ並行処理も行いながら管理しています。 開発が進むにあたって機能を追加/修正してコード量が増加していくと、こういった非同期処理や並行処理が複雑に絡み合うようになっていきます。こうしたコードは可読性やメンテナンス性が下がり、本来枠組しだければいけない処理が待たれずに次の処理へ進んでしまったり、受け取れないはずの操作が状態管理の不備によって受け付けられてしまったりと、意図しない不具合を混入させやすい状態になります。 今回はこういった課題を解決していく手法としてのコルーチンの活用方法について、プロジェクトの品質向上の視点からお話しします。	・コルーチン自体は知っているが、どういった場面で有効なのかより具体例を知りたい方 ・非同期処理や並行処理での状態管理に悩まされている方	25 minutes	日本語 / Japanese	Kotlin	horitamon
Why Projects Succeed: Lessons Learned from the Android OS	The story of how Android came to be, and how it managed to survive and thrive in a crowded field of competitors, is an interesting story of people, teams, and vision. But it's also an educational tale of how products can succeed, which can help inform future ideas, startups, companies, acquisitions, and projects throughout tech and business overall. This session will draw from the recently published book, Androids: the Team That Built the Android Operating System, to see what we can all learn from the Android project, which started as two people building a camera OS and resulted in a platform running on more than 3 billion devices today.	No prerequisite knowledge of Android	40 minutes	English	その他 (Other)	Chet Haase, Romain Guy
あらゆる変化を受け入れながら働きつづける 〜 介護・学業編	【DroidKaigi実行委員会より】 本セッションはDroidKaigi実行委員会による招待セッション（※）です。 Note: This is an invited talk from the DroidKaigi Committee. micchieさんは株式会社メルペイのエンジニアリングマネージャーです。一般社団法人Gophers Japanの一員として、プログラミング言語Goに関するカンファレンス「Go Conference」や女性とジェンダーマイノリティの人々があつまるGo言語のコミュニティ「Women Who Go Tokyo」の運営に携わっています。働きながらご家族の遠距離介護を行った経験があり、現在は社会人大学院生として大学院に通っています。 本セッションでは、介護や学業といったプライベートを仕事と両立されたご経験をご講演いただき、ワークライフバランスを考慮した多様な働き方や将来的なキャリアの描き方について考えます。 受講対象者： ・家族のケアや学び直しといったプライベートとの両立に悩んでいる方、視野に入れている方 ・プライベートについてさまざまな事情を抱えるメンバーと共に働いている方 ・マネージメント職を検討している方 ・中長期的な視点からキャリアパスを考えたい方 ※DroidKaigi実行委員会は多様性向上を通して日本のAndroidエンジニアコミュニティのより良い発展に貢献することを目標としています。 DroidKaigi 2022では、働き方やユーザの観点から多様性を考える場を招待セッションのかたちで設けました。参加者の皆様に自身やDroidKaigiコミュニティのあり方について考えるきっかけとなれば幸いです。	・家族のケアや学び直しといったプライベートとの両立に悩んでいる方、視野に入れている方 ・プライベートについてさまざまな事情を抱えるメンバーと共に働いている方 ・マネージメント職を検討している方 ・中長期的な視点からキャリアパスを考えたい方	40 minutes	日本語 / Japanese	その他 (Other)	micchie
アクセシビリティは向上させる物 〜視覚障害当事者の立場で私が目指している理想的な社会〜	【DroidKaigi実行委員会より】 本セッションはDroidKaigi実行委員会による招待セッション（※）です。 Note: This is an invited talk from the DroidKaigi Committee. maverickさんは株式会社SmartHRのアクセシビリティスペシャリストで、先天性の全盲の視覚障害者です。デジタルアクセシビリティをテーマとしたランチ会「Accessiブランチ」を主催し、Windows用スクリーンリーダーの日本語版開発コミュニティ「NVDA日本語チーム」のPR責を務めるなど、アクセシビリティの向上に日々尽力しています。 本セッションでは、当事者の視点を交えながらアクセシビリティエンジニアの会社事と働き方についてご講演いただき、視覚障害者の方々が置かれている状況やアクセシビリティ分野の課題、障害をふまえた働き方について考えます。 受講対象者： ・アクセシビリティ分野に興味がある方、アプリのアクセシビリティを向上させたい方 ・アクセシビリティを専門とするエンジニアとの連携を高めたい方 ・障害のある方から見た労働環境に興味がある方 ・より多くの人々にとって使いやすいシステムやデジタル環境を推進したい方 ※DroidKaigi実行委員会は多様性向上を通して日本のAndroidエンジニアコミュニティのより良い発展に貢献することを目標としています。 DroidKaigi 2022では、働き方やユーザの観点から多様性を考える場を招待セッションのかたちで設けました。参加者の皆様に自身やDroidKaigiコミュニティのあり方について考えるきっかけとなれば幸いです。	・アクセシビリティ分野に興味がある方、アプリのアクセシビリティを向上させたい方 ・アクセシビリティを専門とするエンジニアとの連携を高めたい方 ・障害のある方から見た労働環境に興味がある方 ・より多くの人々にとって使いやすいシステムやデジタル環境を推進したい方	40 minutes	日本語 / Japanese	その他 (Other)	maverick
Camera Xライブラリの魅力と最新機能を紐解く	Camera Xライブラリは、カメラアプリの開発を容易にすることを目的としたJetpackライブラリです。Android 5.0以降を搭載したデバイスをサポートしており、Camera 2ライブラリと比較して圧倒的に少ないコード量で一般的なカメラのユースケースや、デバイスのネイティブカメラアプリと同じ機能を実現することができます。 最新のstable版であるバージョン1.1.0は2022年6月末に公開されました。実に1年ぶりとなるマイナーバージョンのアップデートであり多くの新機能が加わっています。 本セッションでは、一見複雑多岐に見えるCamera Xライブラリを分かりやすく紹介しつつ、バージョン1.1.0の新機能を始めとした様々な魅力をお伝えすることで、カメラ機能を生かした新たなアイデアを生み出すきっかけとなることを目指します。 発表予定の目次 ・Camera Xライブラリの概要、基本的な実装の解説 ・バージョン1.1.0での新機能の解説・実演 ・バージョン1.1.0ではまだ実現できず課題を抱えていることの紹介	・カメラ機能をアプリに組み込みたいと考えている人 ・現在Camera 2ライブラリなどを使っている人 ・Camera Xライブラリを使うことでどのようなことができるのか興味を持っている人	25 minutes	日本語 / Japanese	ハードウェア (Hardware)	DaiMiyamoto
Optimize your app for large screens	日本市場でも、タブレットや折りたたみ式デバイスがリリースされてきています。これらのデバイスでは、アプリはスマホと比べて大きなスペースを利用することができます。また複数のアプリを同時に使うマルチタスク機能も、スマホと比べて多く利用されることもわかってきています。 このセッションでは、タブレットや折りたたみ式デバイスのような大画面デバイスをサポートするにあたって考慮しなければならないこと、そしてサポートするための技術的な選択肢について紹介します。具体的には Jetpack Window Manager や、SlidingPaneLayout、Activity embedding のような大画面デバイスをサポートするために有用な API、レスポンシブレイアウトを実現するための方法、大画面デバイスをサポートする際に利用できるテスト手法について紹介します。	Android アプリ開発にある程度の経験をお持ちの方。Android アプリ開発の基本的な知識（例えば Activity のライフサイクルなど）を前提としたセッションです。アプリ開発の基本的な知識をお持ちでない方は、セッションを楽しんでいただけない可能性が高いように思います。 セッションで表示するサンプルコードを Kotlin で記述します。Kotlin の基本的な文法を知っている方であれば、よりセッションを楽しんでいただけると思います。	40 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Chiko Shimizu
サービス成長のためのイベントトラッキング戦略	自社のサービスを成長させようとしたき、姿の見えないユーザーがどんな利用をしてくれているのかを知るにはどうするとよいでしょうか？ 解決手段の一つが、イベントトラッキングを行うことです。 本セッションでは、Androidアプリを利用するユーザーのイベントトラッキングの必要性と、取得ロジックの設計、そして運用としてどのように分析を行うか、イベントが誰って消されないようにするための予防措置を弊社がどのように実現しようとしているのかをお話します。	- 業務でサービス成長のためにイベントトラッキングを行おうとしている方 - 現在イベントラッキングを行っている方で、自社の収集方法に課題を感じていて、他社の事例を知りたい方	40 minutes	日本語 / Japanese	開発体制 (Development Process)	shige0501

Androidエンジニアのためのfigma入門	Material Design 3では、デザイントークンの項目が大きく刷新され、
	昨年のAndroid Developer Summitでは、Figma上のUIからJetpack
	Composeのコードを出力するワークフローの予定が明かされました。
	
	これらのトレンドが示すのは、アプリデザインの領域が、絵を描き起こす
	作業から情報設計にシフトしていることです。デザインとエンジニアリング
	の境界が曖昧になったとき、アプリエンジニアがデザインツールを熟知す
	ることは、今後必要不可欠になるかもしれません。
	
	本セッションでは、こうした時代に備えて、Figmaの基礎を広く浅くなぞり
	つつ、Figmaのデザインシステムに関連した機能やMaterial Design 3と
	の関連を紹介していきたいと思います。Androidエンジニアが語る、
	Androidエンジニアから見たFigmaの要点、というのがポイントになりま
	す。
	
	アジェンダとしては、以下を予定しています。
	
	- 基礎
	  - 基本操作
	  - Team, Project, File
	  - Layer, Frame, Group, Page
	- デザインシステム
	  - Component, Style, Library
	  - Variants
	  - デザイナとエンジニアの協業、Tips
	- FigmaとMaterial Design 3
	  - Design to Code...?
	  - Material 3 Design Kit
	  - Material Theme Builder
	  - Material Symbols
		- 雰囲気でFigmaを使っている人
		- 普段見ているFigma上のUIが、どういった機能を使って作られているの
		か知りたい人
		- デザインの話が好きなエンジニア
		- 情報設計に関心があるデザイナー
			25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	haru067

Android プロジェクトで大規模リファクタリングを安全に遂行する	Android (Jetpack) の API はバージョンが上がるたびにめまぐるしく更 新され、いつの間にか API が非推奨になっていたり新しい API が作られ たりします。新規プロジェクトや小さなプロジェクトにおいて対応するのは 容易ですが、そこそこ既存のコードの規模が大きい場合は大変です。手 動で対応しようとすると時間がかかるだけでなくミスが起きやすくなりま す。そこで 迅速かつ安全に最新の API へのマイグレーションを行う手法 について議論します。 本セッションでは以下の3つのレベルに分けて具体的な Android (Jetpack) の API のマイグレーション例を交えながら紹介します。 1. 一般的な置換によるマイグレーション 2. Android Studio の Structural replace による置換によるマイグレーション 3. Kotlin の構文解析器をつかった高度な置換によるマイグレーション (Koin→Hilt)	* 新しい API にマイグレーションしたいが、規模が大きく手が出ていない 方 * 手動で API のマイグレーションを行ってきた方	25 minutes	日本語 / Japanese	Jetpack	Naoto Ishida
Jetpack Compose の空白をマスターする	空白はデザインの重要な要素の 1つ です。Jetpack Compose では空 白を柔軟にコントロールできるようになりました。しかしながら、どの API で空白を実現するか悩んだり うまく空白をコントロールできないケースがあ ります。本セッションでは Jetpack Compose における 空白の実現方 法とベストプラクティスについてできるかぎり網羅的に紹介します。 Spacer Modifier.padding Modifier.weight PaddingValues (contentPadding) Arrangement Font padding Insets Layout	Jetpack Compose において * Spacer のみで大雑把に空白を調整している人 * 色々な空白の実現方法を知りたい人 * UIの構造を踏まえて空白を設定したい人	25 minutes	日本語 / Japanese	Jetpack Compose	Naoto Ishida
Android アプリのリリースフローをできる限り自動化する	Android アプリを運用する上で欠かせないリリース作業ですが、リリース 後の運用を考えたりすると「リリースビルドして aab を生成しアップロード する」以外の手順が増加しがちではないかと思います。例えばリリースした 日を管理するためにカレンダーに登録したり、git flow にあるようなブラン チ運用をしたり、 そういった手順を手作業で行うとなると、いくらチェックリストを作って確認 などをしていても、人間なのでミスをしてしまう可能性があります。 このセッションでは様々な API などを多用してリリースフローにおける手 作業を「できる限り」減らすことを目指し、自動化の仕組みを実現する方 法を紹介します。 <キーワード> - Google Play Developer API - GitHub API - Bitrise, GitHub Actions（ビルドする基盤）	リリースフローを自動化したい人 リリースフローを手作業でやっていて困っていないが事例を知りたい人	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	bird_tummy
UIカタログのすゝめ	ウェブやモバイルなどフロントエンド開発においてはUIを実装する部分 は、エンジニアが特に注力するポイントになります。ウェブフロント開発に おいては、デザインシステムのUIコンポーネントの実装を確認できるよう にStorybookというUIカタログツールを導入することがデファクトスタン ダードとなってきています。 一方でAndroidアプリ開発においては Jetpack Compose の登場によっ て、ウェブフロントからの概念が輸入されているトレンドになっており、 Storybookのようなカタログツールも徐々に台頭してきています。 当セッションでは UIカタログ を導入するメリットから簡単な導入方法につ いて解説します。	* UIカタログをこれから導入したい方 * デザイナーフレンドリーな開発をしたい方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	kgmyshin
定義へGO! Jetpack Compose の基本 API の実装から学ぶ	Jetpack Compose のコードを書いていると毎日のように使う Button、 Column、Modifier 等がどのように実装されているかご存知ですか？ 普 段使うAPI よりも一段低いレベルのレイヤーの実装を覗いてみると、驚く ほど多くの発見と学びがあります。一種にコードリーディングとして学び ませんか。主に androidx.compose.foundation / ui / material のコード が対象です。 * Button * Column/Row * Box * Surface * Modifier	Jetpack Compose を一段低いレイヤーで理解したい人 Jetpack Compose で普段使う API の実装を知りたい人	40 minutes	日本語 / Japanese	Jetpack Compose	Naoto Ishida
パフォーマンスを意識したJetpackCompose実践	JetpackComposeではKotlin APIを採用することで従来のViewシステム よりも自由度が高くUI周を記述することができるようになりました。 しかし、これにより意識せずに構築するとパフォーマンスが上がらない実 装になってしまうことがあります。 本セッションでは、筆者が実務で導入する際に実装した経験を元にパ フォーマンスよく実装するための例を実載形式でお伝えできればと思い ます。 アジェンダ(予定) ■ Jetpack Composeでのパフォーマンスを上げる方法について はじめに、JetpackComposeでパフォーマンスを上げるための一般的な アプローチの方法を解説します。 - 計算ロジックのViewModelへの分離 - 不変性を意識したデータ設計 - 動的な変更区分を意識したCompose関数の分離 - 不要なイベント発行やRecomposeを起こさないためのビューの状態監 視 ■ 実践例 ユーザ操作可能なアニメーション付きのローテーションバナー ローテーションバナーを例に時間経過で更新されるアニメーションとユー ザスワイプなどによるイベントハンドリングをJetpack Composeを 用いてパフォーマンスよく実装する方法を実践例を用いて解説します。 ■ 実践例 ボトムシートを用いた絞り込み・ソート有りのページングリスト 絞り込み・ソートやスクロールによる追加読み込みなど動的な操作が可 能なページングリストをJetpack Composeを 用いてパフォーマンスよく実装する方法を実践例を用いて解説します。 ■ 実践例 複数の状態をマージしてのエラーダイアログやエラー画面 エラーダイアログやエラー画面では、複数のAPIレスポンスやシステムの 状態を監視して結合して掲出する必要があります。 これらをJetpack Composeを用いてパフォーマンスよく実装する方法を 実践例を用いて解説します。	これからJetpackComposeを実務で導入することを検討している方 JetpackComposeを導入したが、いまいちパフォーマンスが上がらずに 悩んでいる方	40 minutes	日本語 / Japanese	Jetpack Compose	Yukihiro Mori

AndroidアプリにFlutterを載せる	あなたのアプリにもFlutterを導入してみませんか？
	
	既存のモバイルサービスではAndroidやiOSそしてWebまで別の開発
	チームが必要になり、コミュニケーションコストやプラットフォームによって
	差分が発生することが多いと思います。
	
	そして会社的にはAndroid、iOS、Webまで別々のチームを作って数人
	で維持することは難しいと思います。
	
	上記の問題を解決できるクロスプラットフォームを提供するFlutterがあり
	ますが、一気に変更することは現実的ではないのでネイティブから順様っ
	て開発している会社もあると思います。
	
	そこで、既存アプリの一部をFlutterで変更することはどうでしょうか？
	少しでも上の問題を解決できるし、少しづつFlutterを学習することもでき
	ると思います。
	
	弊社では既存サービスにFlutterをのせることを検討していますが、検討
	しながら得た情報を皆さんと共有したいと思います。
	
	概要
	＊ 既存AndroidアプリにFlutter載せる方法
	＊ 載せたFlutterとのデータやりとり
	＊ 既存サービスにFlutterを載せてみて導出した結論
	＊ 各AndroidとFlutterチャンネルの性能
	  - EventChannel移動時間
	  - EventChannelの限界のデータ量
	  - MethodChannel移動時間
	  - MethodChannelの限界のデータ量
		Androud開発者	25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	Tomi Hand

Gradle Managed Virtual Devicesで変化するエミュレータ活用術	皆さんはAndroidエミュレータを使っていますか？ 普段の開発であたり前のように使われているエミュレータですが、Gradle Managed Virtual Devicesの導入によって、Androidエミュレータを取り巻く事情が大きく変化しようとしています。 Gradle Managed Virtual Devicesは、Android Gradle Plugin (AGP) 7.2より導入された機能です。 この機能を使うと、build.gradleにテストを走らせたいAndroidエミュレータのスペックを書くだけでAVD (Android Virtual Device)の作成・起動からテストまでをコマンド1つで実行できます。 また、(Canary版ではあるものの)Automated Test Device (ATD)と呼ばれる軽量なエミュレータも利用できるようになっています。 一見とても便利そうなGradle Managed Virtual Devicesですが、本機能にも得手不得手があります。 例えば、画面が表示されなかったり、テスト実行が終わると同時にエミュレータも終了したりするため、意図通り動かなかったときの解析が大変なことがあります。 本セッションでは、新しく導入されたGradle Managed Virtual Devices機能はどのようなものなのか、それによってエミュレータの使い方がどのように変化してくのか紹介します。 また、AGPの実装を調べて判明した内容を元に、Gradle Managed Virtual Devicesを活用する際に直面しがちなトラブルの解決方法も合わせて紹介します。 具体的には次のようなトピックなどについて紹介します。 これを機に、皆さんが持っているAndroidエミュレータの知識もアップデートしませんか？ ・Gradle Managed Virtual Devicesでできること、できないこと ・Gradle Managed Virtual Devicesの活用例 ・Robolectricでは心許無いテストを動かす ・UIテストを動かす ・スクリーンショットテストを動かす ・CI環境上でGradle Managed Virtual Devicesを使うときの工夫 ・Gradle Managed Virtual Devicesの実行で問題が起きたときの調査方法Tips ※できるだけCanary版を含む最新のAGPの実装を調べた結果にもとづいて話す予定です。	・Gradle Managed Virtual Devices機能に興味はあるが、深く触っていない方 ・Gradle Managed Virtual Devices機能を試してみたものの、思った通りに動作せず諦めてしまった方 ・CI環境において、Androidエミュレータ上でテストを走らせる方法を知りたい方	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	外山純生 (sumio_tym)

Introduction to Wear OS Application Development	みなさま、ウェアラブルデバイスはお使いですか？
	Google I/O 2022で、今秋にGoogle Pixel Watchが発売されることが公表されました。
	
	Wear OSの開発をめぐる環境は、近年大幅にアップデートされています。
	たとえばJetpack APIの登場や、Android Studio連携の強化、ドキュメントやコードラボの刷新などが行われました。
	
	本セッションでは、Wear OSアプリのリリースを目標に、これを知っていればWear OSのアプリを実装できるという要点を紹介します。特に、Wear OSアプリの開発をこれから始められる初心者の方を想定し、Wear OSがどのようなもので、どんな機能が提供できるのかという点も詳しく解説します。
	
	現在考えているセッションの内容は次のとおりです。
	- Wear OSの開発について
	  - Wear OSではどんなことができるのか
	  - スマートフォンのアプリ開発との差異
	  - Wear OSアプリのUIとUX
	- APIのサポート状況の概観
	  - Jetpack API
	  - Compose for Wear OS
	- Wear OSの開発の始め方
	  - セットアップ
	  - デバッグ方法
	- Wear OSアプリをつくる
	  - Overlays
	  - Tiles
	  - Complications
	
	発表をとおして、最新のWear OSアプリの開発情報について共有できれば幸いです。
		・Wear OS向けのアプリに入門されたい方
		・以前、Wear OS向けのアプリを開発されていた方
		・最近のWear OS向けのアプリの開発情報をキャッチアップされたい方
			40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Ryo Yamazaki

ノーアーキテクチャなコードにアーキテクチャをもたらす技術	往々にして、ビジネスとしてのサービス開発初期からアーキテクチャを導入するのは悩ましいものです。数年後どれだけスケールしているのか？そもそも当たるのか？というのが定かではない状態でそこにコストを掛けるよりも、実際に動くものをユーザーに最速で提供することで高速なPDCAを回していきたいからです。いわゆる0→1のフェーズを突破して、1→10のフェーズに到達したサービスが抱える問題のひとつに開発初期のノーアーキテクチャなソースコードの運用が挙げられます。 ありがたいことに、アプリアーキテクチャガイドというものがGoogleから公開されています。Androidアプリエンジニア同士の共通認識として活用できる資料として活用されている方も多いでしょう。 本セッションでは、ActivityやFragmentなどにロジックがベタ書きされているノーアーキテクチャなソースコードにアーキテクチャを導入するためのテクニックその過程、継続的に運用していくための取り組みをご紹介します。当時の失敗談も交えながら、今だったらこうしていたという改善案まで赤裸々にお話する予定です。	- ノーアーキテクチャなソースコードに苦戦している方 - Googleのアプリアーキテクチャガイドに興味がある方 - 「関心の分離」「レイヤードアーキテクチャ」「DI」でピン！とくる方 - 技術的負債でお困りの方 - 担当サービスが0→1から1→10にフェーズ移行される方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	kobaken
Androidのモダンな技術選択にあわせて自動テストもアップデートしよう	ここ数年の間に、Jetpack ComposeやKotlin CoroutineおよびKotlin Flow等、Androidアプリ開発をする際に選択できる新しい技術が登場しました。現在では、多くのプロダクトがそれらの技術の導入や移行を進めています。 新しい技術を選択する際には当然キャッチアップが必要になりますが、あわせてその技術を使ったコードをどのようにテストするかを知ることも重要です。テストする手段を知ることで、新しい技術を使ったコードの動作確認も容易になりますし、その技術を更に理解することにも繋がります。とはいえ、プロダクトコードへの導入を進めながら、テストの書き方までキャッチアップするのが大変なことも事実です。 本セッションでは、新しく採用されるケースが多い、次の技術のテスト方法を紹介します。また、より実践的な内容にするために、テスト方法をGuide to app architectureで推奨されるレイヤー毎(UI Layer・Data Layer)に整理して紹介していきます。 ・Kotlin Coroutine ・Kotlin Flow ・Jetpack Compose ・Dagger Hilt 扱うトピックが幅広いため、1つ1つのテスト方法について深堀りするよりも、それぞれの技術・レイヤーの最初のテストを書くために必要な知識を整理し、テストを書き始めるハードルを下げることを目指します。	・Kotlin CoroutineおよびKotlin Flow/Jetpack Compose/Dagger Hiltの導入をしたが、自動テストのキャッチアップができていない方 ・Kotlin CoroutineおよびKotlin Flow/Jetpack Compose/Dagger Hiltを導入予定で、自動テストの基本的な書き方に興味がある方	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Nozomi Takuma
なぜあなたのデザインは正しく実装されないのか	あなたが心血を注いで作り上げたデザインは、エンジニアの実装を通した結果、なんだか思ってたのと違う形でアウトプットされてしまいました。なぜ、このような悲しいことが起きてしまうのでしょうか。 そこにはデザイナー、エンジニア、時にはマネージャー(オーナー)のそれぞれに何か原因があったり、時には全員が同じ問題を抱えている場合もあります。 本セッションでは、ありがちな「意図通りに実装されないデザイン」の例を上げながら、なぜそうなってしまうかという原因を分析しつつ、意図通りに実装されやすいデザインはどういうものかを学んでいきます。 なお、Kotlinによる非ゲーム系アプリを開発することを前提としたセッションとなります。	基本的にはデザイナーが受講対象者ですが、デザイナーにデザイン内容のやり取りで悩みを抱えているエンジニアやマネージャーの方も参考になるかもしれません。	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	yamacraft
着信から応答まで	通話に関する機能を提供するTelecom frameworkを使い、VoIP通話アプリでどのように着信から応答まで実装するか	アプリが終了している状態での着信処理に興味がある人	25 minutes	日本語 / Japanese	Android Framework	retlat
BLEを使ったアプリを継続的に開発するために	BLE、Bluetooth Low Energyとは近距離無線通信規格Bluetoothのうち低消費電力で利用できる通信モードで、IoTデバイスとスマートフォン間の通信などで広く利用されています。 AndroidというOSにおいては、端末依存の挙動やコールバック地獄などが存在するおちゃめなAPIとしても一部で名が通っています。 IoTデバイスを開発している企業にAndroidエンジニアとして入社した場合、BLE APIと付き合っていくことになります。現在提供されているコールバックベースのBLE APIを受け入れて真摯に向き合っていくのはおすすめできません。RxJava時代を経て今我々の手にはCoroutineという素晴らしい道具があります。このセッションではBLE APIを利用したデバイスとの通信の実装を解説した後に、私が経験したCoroutineを利用したメンテナンス性の高いコードへの書き換え事例やプラクティスを紹介します。 # こんな人に聴いて欲しい - BLE APIを利用するアプリを開発していて、コードのメンテナンスに疲れてしまった方 - これからBLE APIを利用するアプリを書こうとしている方 - メンテナンス性の高いBLE APIを利用するコードを書きたい方 # 話すこと - BLEに対応したデバイスとの通信を実装する方法 - 2022年、Coroutine/Flow時代のBLE APIプラクティス # 話さないこと - プロトコルの詳細 - BLE APIのおちゃめさ加減 # こんな人を待ってます - BleXを開発してくれるGoogler	BLE APIを利用するアプリを開発していて、コードのメンテナンスに疲れてしまった方 これからBLE APIを利用するアプリを書こうとしている方 メンテナンス性の高いBLE APIを利用するコードを書きたい方	25 minutes	日本語 / Japanese	Android Framework	Moyuru

今更覚えるRecyclerView 第一話	Jetpack Composeが隆盛を極める昨今、legacyとなってしまったViewに関するセッションです。 多量のアイテムを表示するのに使用するといえばJetpack ComposeのLazyColumnやLazyRowでしょうが、以前はRecyclerViewを使用していました。今となってはlegacyなRecyclerViewですが、UIの複雑さによってはLazyColumn/Lazy RowよりRecyclerViewの方が高いパフォーマンスを維持しやすく、まだ十分な利用価値があるでしょう。 RecyclerViewは様々なコンポーネントから構成されており、そのコンポーネントをカスタマイズすることでただアイテムを並べるだけでなく様々な表現やパフォーマンスの向上が可能になっています。代表的なものです以下のようなコンポーネントがあります。 - RecyclerView.ItemDecoration - RecyclerView.LayoutManager - RecyclerView.ItemAnimator - RecyclerView.Adapter - RecyclerView.RecycledViewPool - SnapHelper - ItemTouchHelper - DiffUtil 例えば、RecyclerView.ItemDecorationをカスタマイズすればアイテム間にフレキシブルにマージンを付けることができ、応用すればDroidKaigi 2019アプリのセッション一覧画面のようにアイテムの横に時間を表示することも可能です。 RecyclerView.LayoutManagerをカスタマイズすれば同じくDroidKaigi 2019アプリのタイムテーブル画面のような複雑なレイアウトを実現できます。 RecyclerView.RecycledViewPoolやDiffUtilを活用することでパフォーマンス向上を図ることも可能です。 このように、RecyclerViewは非常に拡張性の高いViewです。 いつか来るであろうデザイナーからの高い要求に「出来ます」と余裕の一言を返すために、本セッションでは以下の2つを通してRecyclerViewの理解を深めましょう。 # RecyclerView.ItemDecorationでCanvas直描き装飾 前述したようにItemDecorationはアイテム間にマージンをつけるだけにとどまりません。RecyclerViewのCanvasにアクセスすることでRecyclerViewのアイテム間を縦横無尽に装飾する事が可能です。 # LinearLayoutManagerをミニマム再実装してRecyclerView.LayoutManagerの実装理解する RecyclerView随一の複雑かつ重要なコンポーネント、LayoutManagerの実体方法をLinearLayoutManagerのミニマム実装を通して理解しましょう。 * その他のコンポーネントに関してはFuture DroidKaigiで!!	- まだまだRecyclerViewを使っていきたい奇特な方 - 複雑なタイムテーブルが組まれてしまった年のDroidKaigiのアプリ開発者や某TVの開発者	40 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Moyuru
アプリ改善〜レガシーとモダンの共存と移行〜	Android アプリを開発しながら継続してリリースしていく上で所謂レガシーと呼ばれる技術とモダンと呼ばれる技術をどう使っていくかは大きな課題だと思います。 本セッションでは、レガシーからモダンまでが共存しつつ、移行しながら継続的にリリースするアプリの開発に携わっている経験から、どのように考えて進めているか、良かったこと悪かったこと等の知見を共有できたらと思います。 ※ここでのレガシーは絶対悪という文脈では使っていません。 セッションに出てくるであろう技術 レガシー - Java - AndroidAnnotations - EventBus 等 モダン - Kotlin - Jetpack Compose - Dagger Hilt - 各種 Jetpack ライブラリ 等	- レガシーな Android アプリをどう変えていこうか悩んでいる方 - レガシー → モダン絶賛移行中の方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Pon
今年リリースされたCTS-D (互換性テストスイート)をアプリ開発に活用しよう！	Androidアプリを開発していて、あるデバイスでは動作するのに、他のデバイスでは、予想通りに動作せず、苦労したという経験はありませんか？ このセッションでは、デバイスの互換性をテストするCTS-Dについてご紹介します。CTS-Dは、グーグルが今年6月にリリースしたプログラムで、アプリ開発者の方に、互換テストの共同開発に参加していただき、互換性の問題を、Androidエコシティーとして、一緒に解決していくことを目指して発足されました。	Androidの開発者全般。初心者〜上級者。 CTS-D（互換性テストスイート）の基本から、テスト開発への参加方法までを網羅します。	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	ctsd, Saryong Kang
State of Jetpack Compose	Hear about Jetpack Compose from the Compose team. What's new in Compose 1.2, what's coming in 1.3 and tips you can use to adopt Compose in your app.	All levels of Android developer	40 minutes	English	Jetpack Compose	Ben Trengrove
Lessons Learned Migrating the Maps SDK to Compose	This session goes over the story behind Maps Compose—the Jetpack Compose interoperability library for the Maps SDK for Android. Learn lessons in Compose API design and how to effectively migrate your library or app to Compose. At the end of this talk, developers will know how to: * use interoperability APIs to bridge View-based apps/libraries to Compose * follow best practices for composable elements' API design * use subcomposition to tap into Compose APIs for managing tree-like data structures	Intermediate developers. Have prior knowledge of Jetpack Compose.	40 minutes	English	Jetpack Compose	Chris Arriola
KoinからDagger Hiltへの乗り換えで見えた安全な移行手段	Dependency Injectionは依存関係の分離を行うことでリファクタリング・テストを容易にするなど、よりよいアプリアーキテクチャのためにAndroidアプリ開発でも広く導入されています。 これまではDIライブラリとしてDaggerやKoinが存在していました。しかしDaggerは概念が複雑でボイラープレートが多くなってしまったり、KoinではDIの設定のミスがビルド時ではなくランタイム時にでてしまうといった問題がありました。 そんな中登場したのがDagger Hiltです。 弊社アプリでは半年をかけてKoinからHiltへの移行作業を完了させました。 DIはアプリ全体に使われるものであり、移行作業の影響範囲はアプリ全体になります。とはいえいきなり移行することは難しいので、アプリの他の機能開発を続けながら安全にかつなるべくコストを抑えて移行する手段を選択しました。 移行作業を行う中で、制約に厳しいHiltでは直接表現できず修正が必要な箇所が出てきました。ここから、Hiltで禁じている「良くない設計」も見えてきました。 このトークでは、実際に移行を行った知見をもとに ・DIとは ・各DIライブラリの共通する・異なる機能 ・開発を続けながら大規模な移行を安全に行う方法 ・Hilt化で見えてきた良くない設計 ・ビルドやランタイムでのパフォーマンス比較 について話します。	KoinからHiltへ移行したい方 DIライブラリの選定を行っている方 大規模な移行作業を安全に行うヒントが欲しい方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	vemo3632
ユーザーの行動ログ定義を Markdown で管理して Gradle Plugin からコードを生成する	ユーザーの行動ログはサービスを開発する上で分析や実態把握に用いられるものです。それらは日々のサービス開発によって多くの施策に対して変異され、実装され、積み重なっていきます。 ログの送信箇所が増えれば増えるほどに、またログを実装してから時間が経てば経つほどに、どの画面でなんのログが送られているか、そのログにはどういう情報が乗っているかなどを解釈するのが困難になります。 また、iOS/Android で分かれてアプリを開発している場合は、どうしてもイベント名を間違って実装したり送信するタイミングを揃えられなかったりすると、分析をするときに多くの手間や手戻りが発生してしまいます。 それらの問題を解消するために、Markdown でログの定義を共有し、そこから Gradle Plugin を使ってログのコード自動生成を行う仕組みを導入しました。 本セッションでは、導入に際して私たちが持っていた課題を共有し、それらの課題に対してどう向き合ったかを紹介した上で、Gradle Plugin からコードを生成するときのメリット、kapt, ksp との違い、実装時のポイントなどの技術的な取り組みについて紹介します。	行動ログのドキュメンテーションや管理方法に困っている人 iOS/Android でプラットフォームをまたいでログの定義を管理するのに疲れた人	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	litmon
エンジニア駆動でデザインツールの刷新ができた	昨今では、アジャイル開発でのスタイルが主流となりつつある中で、デザイナーとエンジニアは密なコミュニケーションを取りながら要件決め・プロダクトの実現性や設計などを一緒に作っていく必要があると思います。 デザイナーとのキャッチボールがうまくいかないと開発スピードが十分に出ない事や要件の認識違いなど致命的な問題に遭遇することもあります。 そういった問題をツールの刷新で解消できたお話をできたらと思います。	エンジニア・デザイナーとのコミュニケーションやインテグレーションで困っている方 ツールなどの刷新をするきっかけに困っている方	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	bami
PlayStoreでの新しいユーザー訴求 -LiveOpsの活用とその成果-	みなさんは自社アプリの魅力をユーザーにどのように伝えていますか？ ストアに掲載する画像の趣向を凝らしたり、イベント時にはアプリアイコンを変えるといったことは定番の手法になったかと思います。 ところで、最近はPlayStoreにアプリのイベントや特典の告知が表示されるようになったことは、ご存じでしょうか。これはPlayStoreのLiveOpsという機能を使って提供されており、新たにユーザーの獲得や既存ユーザーのエンゲージメント向上に役立つ機能となっています。 本セッションでは、LiveOpsで何が出来るのか、実際にどのような効果が得られるのかといった点を中心に、ZOZOTOWNアプリでの実例を交えながら解説してみたいと思います。 想定する内容 ・LiveOpsで扱える内容 ・LiveOpsの設定方法 ・設定時の注意点 ・ZOZOTOWNアプリで設定したLiveOps ・ZOZOTOWNアプリで得られた成果	・アプリのダウンロード数を増やしたいと思っている方 ・アプリのアクティブユーザー数を増やしたいと思っている方	25 minutes	日本語 / Japanese	その他 (Other)	androhi
Android アプリの内と外をつなぐ UI	UI はアプリの中だけのものではありません。ホーム画面からアプリを起動する際のスプラッシュ スクリーン、ステータス バーやナビゲーション バーとアプリ内 UI の干渉を防ぐ WindowInsets など、アプリの UI がシステムの UI と連携する場面は数多くあります。最新の Android 13 でもバック ジェスチャーに関する API が変更され、対応が必要になっています。このセッションでは、アプリの内と外をつなぐ UI に焦点を当て、最新のベスト プラクティスを解説します。	View また Compose でアプリを開発している人	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	荒木佑一
Glance で始めるウィジェット開発	Glance は Compose に似た API でホーム画面用のウィジェットや Wear OS のタイルを作ることができるライブラリです。Compose の利用が広がる中、ウィジェットも同じように記述できる Glance の導入を検討している方も多いようです。このセッションでは、Glance でウィジェットを作成する基本的な手順を紹介した上で Compose との違いやよく利用する API について解説します。	ウィジェットの開発を検討している方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	荒木佑一

大規模アプリでのJetpack Composeの導入事例	Jetpack Compose、流行ってきてますよね。 うちも入れてみたいけどな〜、どこから入れるといいのかな〜とお悩みのあなた！ ZOZOTOWNでJetpack Composeを導入した際に得た知見を共有します。 以下のような観点をお話します。 - 導入するためにどのような準備を行ってきたのか - どこから導入をすればよいか - Compose、どういう粒度で分割したらよいか - アプリを作る上でどういった体制で行ったか - Composeを導入する際に何に苦労したか - 参考になったOSS - 得た知見からの改善案	Jetpack Composeを新規で導入したいエンジニア 大規模アプリを開発しているエンジニア	25 minutes	日本語 / Japanese	Jetpack Compose	iwata_n
ラクに速く正確にUIが作れちゃう！Jetpack Composeを用いたデザインシステムによるアプリの開発事例	▼概要 宣言的UIフレームワークのJetpack Composeの登場により、以前に比べてUIコンポーネントが作りやすくなりました。 そこで弊社ではJetpack Composeの特徴を活かし、社内のデザインシステムをパッケージとして開発・運用することで、楽に速く正確にUIを実装することを可能にしました。 また、パッケージ内のコンポーネントを一覧で確認できるカタログアプリも合わせて開発することで、実装コストだけでなく、仕様検討時のコミュニケーションコストやデザインの手戻りも削減することができました。 本セッションでは、デザインシステム構築までの開発プロセスと実際に運用してみて得られた思慮についてご紹介します！ デザインシステムを構築し、みんなで楽に速く正確にUIが作れる世界線へ！ ▼コンテンツ ・デザインシステムについて 　・概要と構築の経緯 　・実装方法と運用事例 　・構築して得られた思慮 ・カタログアプリについて 　・概要と活用事例 　・開発して得られた思慮	Jetpack Composeによるデザインシステムの構築に興味がある人	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	yoshikei
アプリアーキテクチャを明文化しチームの開発効率をアップ	現在の Android 開発において、Android Developサイトでも推奨されているように MVVM アーキテクチャを採用した実装スタイルが採用されることが多く、開発者の多くの方が Develop サイトから学習し開発を進めていると思います。しかしながら、弊社のモバイル開発チームは、2020年に組織化された新しいチームであるため、様々な解決するべき課題を抱えております。中でも、 ・技術力向上 ・開発スタイルの習得 ・学習・実装コストの削減 といった課題は、チームの中でも優先度が高いです。そこで、特に優先度が高かった技術的課題へのアプローチとして「アプリアーキテクチャの明文化」を行い、現在新規アプリ開発を進めております。 弊社の開発においも Android 開発で推奨される MVVM アーキテクチャを採用しています。アーキテクチャをよりチームで共通認識とし運用するには、 ・開発体制 ・技術トレンド ・会社文化 を考慮して具体化する必要があります。ある程度経験がある開発者であれば、○○アーキテクチャと聞く大まかなイメージは掴むことはできると思います。しかしながら、アーキテクチャスタイルだけしか決めずに実際に開発を開始すると、詳細設計での認識齟齬が生じ、レビューの度にディスカッションが生じて開発の効率化に影響を与えてくることも少なくありません。逆にもチームでアーキテクチャが明文化されていると、技術的課題へのアプローチだけでなく、新しいメンバーが加わった際の立ち上がりコストにもアプローチすることができます。 一方で、MVVMアーキテクチャは、RxJavaを中心としたアーキテクチャから近年ではJetpackを中心としたアーキテクチャへと変遷しており、現在も進化しています。またオブジェクト指向やドメイン駆動設計の近年の盛り上がりの潮流も踏まえて、弊社ではこれまでのMVVMアーキテクチャを進化させたアーキテクチャを採用しています。本セッションでは、 ・アーキテクチャガイドラインとして具体的に明文化した内容 ・弊社で採用したオブジェクト指向型のアーキテクチャ を中心に、アーキテクチャを明文化して実際の開発に臨んだ話を紹介したいと思います。	Android 開発者全般	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	akkie76
Android Vitalsのデータを自動監視してビジネス指標を向上させよう	Android Vitalsを使用すると、アプリの安定性やパフォーマンスに関する指標を確認できます。 Vitalsの指標の低下は、ユーザ体験の低下に繋がるだけでなく、Playストアでの評価の低下や検索ランキングの低下にも繋がります。 https://developer.android.com/distribute/best-practices/develop/android-vitals?hl=ja GoogleI/O 2022ではVitalsの指標であるANRを50%減らすと、20%のリテンション率向上、30%トランザクション数が向上したプロダクトが紹介されていました。 https://www.youtube.com/watch?v=DYdHLqLVspY したがって、Vitalsの指標はUUやCVRなどのビジネス指標にも影響する可能性が高いため、指標の改善・維持はプロダクトにとって重要な役割となります。 しかしながら、Vitalsの指標を注意深く監視できていなかったり、監視の自動化が難しいため手動での監視となってしまっているのが実情ではないでしょうか？ 2022年3月にPlay Developer Reporting APIが公開され、Vitalsの指標をAPI経由で取得することが可能になり、監視の自動化や分析をより簡単に行うことが可能になりました。 https://android-developers.googleblog.com/2022/03/play-developer-reporting-API.html 本セッションでは、監視の自動化方法のみならず、アラート発生時の弊社の運用方法ついても紹介する予定です。 <アジェンダ(予定)> ・Android Vitalsについて 　・各指標について 　・(Crashlyticsとの使い分けについて) ・Vitals指標を改善するメリットについて ・Play Developer Reporting APIについて ・監視の自動化方法について ・アラートが発生した際の運用方針について ・(よく見かけるクラッシュとその対処方法の紹介) ・導入の効果・知見	・Android Vitalsについて知りたい方 ・Vitals指標の重要性について知りたい方 ・Play Developer Reporting APIについて知りたい方 ・弊社でのVitals監視の自動化方法と運用について知りたい方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Yusuke Suzuki
あなたのアプリにMaterial Designを取り入れる具体的プロセス	Material DesignはGoogleが提唱したデザインシステムであり、Androidプラットフォームとの整合性が取れるように、AndroidアプリはMaterial Designのガイドラインに従うことが推奨されています。 https://material.io/blog/why-we-recommend-material-design-components-android さらにGoogleI/O 2021では、Material Design 3の発表があり、Material Youによるパーソナライズ機能の強化されるなどアップデートが続いています。 Material Designのアップデートは続いている一方で、プロダクトへの導入はなかなか進んでいないのが実情なのではないでしょうか？ その要因としては下記があげられると思います ・エンジニア・デザイナー双方のMaterial Designについての知識不足 ・プロダクトのどの部分に改善ポイントがあるか明らかではない ・実装時間の確保が難しい 弊社では上記要因を取り除くために、下記取り組みを行ってきました。 要因1について、デザイナーとエンジニアで「Material Design勉強会」と呼ばれる勉強会を行い、ガイドラインを輪講形式で学ぶ取り組みを行っています。 要因2については、デザイナーとエンジニアで改善ポイントの提案会を行い、プロダクトのどの箇所にMaterial Designを取り入れることができそうかを議論しています。 要因3については、「Material Design 実装会」と呼ばれる会を行い、提案会で出た案をペアプロ形式で実装する取り組みをしています。 本セッションでは、上記取り組みの詳細とその効果・知見について紹介する予定です。 <アジェンダ(予定)> ・Material Designとは？ ・Material Design勉強会について ・改善ポイントの提案会について ・Material Design実装会について ・効果・知見 ・リリース内容の紹介	・Material Designをプロダクトに取り入れたいが、進め方がわからないデザイナー・エンジニア	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Yusuke Suzuki
Flutter製Androidアプリの UIを向上させるための注意ポイント	FlutterはGoogleによって開発されたマルチプラットフォームアプリケーションを作成するためのフレームワークです。 Flutterを用いてアプリを開発することで効率よく複数プラットフォームのアプリケーションを開発することができます。 一方で、各プラットフォームには特有のUIがあります。 そのUIに沿って開発することでアプリのUXが向上します。 Androidアプリも例外ではありません。 本セッションではFlutterを用いて作成したAndroidアプリをプラットフォームに沿ったUIにするために注意すべきポイントを共有します。	Flutterを用いてアプリ開発をしている方 Flutterの導入を検討している方	25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	yoppie
Kotlin Script Introduction	Various ways to use Kotlin scripts, locally as a Python/Ruby alternative or in the CI, to automate tasks such as a release process. It will explains pros and cons of using Kotlin Script compared to other scripting languages.	Kotlin intermediate with an interest in scripting and/or CI automation	25 minutes	English	保守・運用・テスト (Maintenance, Operations, and Testing)	Pierrick Greze
Kotlinize your CI, from Gradle plugins to Kotlin Script	This talk will explain how I leveraged Kotlin in various ways to write as much automation code as possible with this language, and so being easy to maintain for any Kotlin user. This will introduce several features of Kotlin (like Kotlin script) or Gradle (like Composite Builds, the JavaExec task, Gradle Plugins, etc)	People interested to know more on the CI side of their project (Android or not), or already doing it but facing issues with maintainability due to the various languages used in 1 project (Bash/Python/Ruby/etc).	40 minutes	English	保守・運用・テスト (Maintenance, Operations, and Testing)	Pierrick Greze

Compose for Desktopで始めるAndroid開発効率化ツールの作成	Compose for DesktopはJetbrainsが開発するデスクトップアプリ向け（macOS、Linux、Windows）のUIフレームワークです。Compose for DesktopはJetpack Composeの仕組みを土台にしており、Jetpack Composeと同じ仕組みでデスクトップアプリを作成できます。そのためJetpack Composeを習熟したエンジニアであればCompose for Desktopにて簡単にデスクトップアプリを作成できるようになっています。 というようにCompose for DesktopはAndroidエンジニアにとって学習コストが低く扱いやすいデスクトップアプリ向けのUIフレームワークですがAndroidアプリ開発における使い所がないなという認識の方も多いのかなと思っております。なので本セッションではCompose for DesktopとADBを組み合わせたツール開発の事例を取り上げ、Androidアプリ開発でどのように活用できるか解説をしたいと思います。また今後利用を考えている方のためにCompose for Desktopを利用した開発の良さや辛さなどもあわせて取り上げようと思います。	Compose for Desktopを利用したツールの作り方を知りたい方	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	y_katsuragawa
Jetpack ComposeによるUI実装をAtomic Designで管理	Jetpack ComposeはGoogleによって開発されたUIを実装するためのツールキットです。 AndroidのViewを用いたUI実装とは違い、宣言的UIとなっています。 小さなUIの部品をComposableとして作成し、それらを組み合わせて画面全体を構築していくのがComposeの特徴となっています。 こういったComposableを意味のある単位で管理するために、Atomic Designを利用することができます。 Atomic DesignとはUIをコンポーネントに分類し、階層化する設計手法です。 本セッションでは下記の内容について解説します。 - Atomic Designを利用するメリット - どういうアプリがAtomic Designに向いているか - それぞれの階層に分類する方法 - Atomic Designを少しアレンジしてアプリに合う設計にする	Jetpack Composeを利用している方 Jetpack Composeを導入予定の方	25 minutes	日本語 / Japanese	Jetpack Compose	yoppie
Kotlin Coroutines Test 2022	Kotlin Coroutinesは2022年のAndroidアプリ開発で欠かせない並行プログラミング技術です。 ネットワーク通信やアニメーション実装などをCoroutineを用いて実装するケースが一般化してきました。 Coroutineを採用した実装が増えるにつれて、Coroutineのテストもより重要度が増してきています。 特に複数のCoroutineを同時に動かすような処理は挙動が理解しづらく、しっかりテストを書いておかないと思わぬバグを引き起こします。 皆さんは、実装したCoroutineの処理をちゃんとテストできていますか？ 本セッションではkotlinx.coroutines 1.6.0で一新されたテストAPIの内容を盛り込み、もう一度イチからKotlin Coroutinesのテストを解説します。 - runTestやTestDispatcherを使ったCoroutineテストの基本 - テストにおいてDispatcherの注入が重要な理由 - Android開発でよく使うCoroutineのテスト実装	- 業務でKotlin Coroutinesを使っている方 - 明日から自信を持ってKotlin Coroutinesのテストを書きたい方 - 複雑なCoroutineのテストを作れるように基本を学び直したい方 - 新しいテストAPIであるrunTestやTestDispatcherの使い方をちゃんと知りたい方	25 minutes	日本語 / Japanese	Kotlin	mkeeda
チームワーク溢れるAndroidコミュニティを目指して	日々進歩しているIT業界で成果を出すためには、一人一人が高いスキルを身に着けているのは大切なことです。 しかし私達は個人の成果よりも、チームワークによってもたらされる成果の方が何倍も大きく美しいことを知っています。 同じチームのAndroidエンジニア達が切磋琢磨し、より成果を出すためにはどうすればよいか。 この課題について多くのチームが試行錯誤しているかと思います。 このセッションでは、私達が日頃から実践しているAndroidエンジニアのコミュニティ活動について紹介したいと思います。 大まかに以下のテーマで紹介します。 ① モブプロによる開発体制の紹介 皆さんもモブプロを一度は耳にしたり経験したことがあるかもしれません。 私達のチームでは、ほとんどの開発をペアプロ・モブプロで行ってきました。 私達が実践しているモブプロの進め方と、長年続けてきたからこそわかるモブプロのメリット・デメリットについて紹介します。 ② "チョルチャタンマ"という取り組みの紹介 チョルチャタンマとは韓国語で切磋琢磨という意味です。 チョルチャタンマはチームのAndroidエンジニア達が集まる場所であり、継続的に切磋琢磨することを目的として活動しています。 これまでもチョルチャタンマを通していくつものアクションが生まれてきました。 チョルチャタンマでの取り組みやこれまでの成果について紹介したいと思います。	・チームワークに興味がある人 ・チームと一緒に成長したいと思っている人 ・モブプログラミングに興味がある人	25 minutes	日本語 / Japanese	開発体制 (Development Process)	真輔 藤田, Takaki Tojo
こだわり特選！Jetpackライブラリ！ ~マイナーだけど使うと便利な味をご賞味あれ！~	Jetpackといえば最近はcomposeばかり注目されていますが、他にも数多くライブラリが存在します。 room✕navigationなどは皆さんもよくご存知で、実務等でもご使用されているかと思いますが、例えばdraganddropというJetpackライブラリはご存知でしょうか？ draganddropは、DropHelperクラスを用いることでドラッグ＆ドロップ機能の実装をとても完結にするライブラリですが、とりあえずAndroid Developersの記事をさらっと見て「今度触ろう」で終わっていませんか？ このような、比較的メジャーではない（と思われる）以下のJetpackライブラリについて、実務で想定されるユースケースを交えてご紹介をしたいと思います。 ・draganddrop ・glance ・startupなどを予定しております。	・Jetpackライブラリ知見の幅を広げたい方	25 minutes	日本語 / Japanese	Jetpack	umsys
Compose Canvas API で想像力を広げよう！	Androidで作りたいビューがあるけど実装のイメージが湧かないな～そんなことありませんか？ このセッションではJetpack Compose Canvasの基礎から実装実例までを紹介して、それぞれが実装したいビューを作成できるように目指します！ Canvasでのはじめの一歩から数学との関係、グラフの作成やアニメーションとの組み合わせなどComposeCanvasに関するいろはを紹介をするセッションです。	Jetpack Compose初心者	40 minutes	日本語 / Japanese	Jetpack Compose	h5masa
androidx.lifecycleの今	androidx.lifecycleは最近では当たり前に使用され安定していますが、今でも日々アップデートされています。特にViewModelについてはCreationExtrasという新しいものが登場していたりもします。 このセッションではandroidx.lifecycleの最近1年くらいのアップデートについて紹介していきます。	最近のandroidx.lifecycleのアップデートが知りたい方	25 minutes	日本語 / Japanese	Jetpack	Kenji Abe
ちょっと待った！ AndroidStudioを効率的に活用できていますか？	AndroidStudioで開発をする際に、あの機能のショートカットなんだっけ？こんな機能があったらいいのにな～ そのような事を思ったことありませんか？ このセッションでは、そんな方々を対象に普段開発で実用できるようなAndroidStudioの機能からショートカット、それからプラグインを紹介をし、"かゆいところに手が届く"を目的としたセッションになっています。 エンジニアとして開発技術面に重点を置いていしまいがちですが、このセッションではアプリ作成をする上で基盤となるIDEツールにスポットライトを当て、視聴者のみなさんのアプリの生産性アップに繋げられたら嬉しいと考えております！	AndroidStudioをIDEとしてされているAndroidエンジニアの方	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	h5masa
考察Photo Picker ～ニーズに合った画像・動画アクセス手法を求めて～	Android 13から新しく「Photo Picker」という機能が実装されました。 Photo PickerはGoogleが推奨する、画像や動画への新しいアクセス手段として提供されており、ACTION_PICK_IMAGESでIntentを発行することで起動することが出来ます。これを用いることで、アプリはユーザにメディアアクセスの許可を求めることなく画像や動画へのアクセスが必要な機能を実装することが可能になります。 しかし、Photo Picker以外にもAndroidプラットフォームにはACTION_GET_CONTENTでIntentを発行するなど、似たようなメディアアクセスの手段が提供されている他、ContentResolverを利用して独自にメディアアクセスの機能を実装する手段も存在します。これらの違いや、使い分けはどの様にすべきなのでしょうか？ 本セッションでは、Photo Pickerを中心に画像や動画をターゲットにしたメディアアクセスについて使い方から紹介し、それらの特徴に基づく使い分けについて考察します。2022年のアプリ開発におけるメディアアクセスの手段を理解し、開発しているアプリの性質に合わせてどれを利用すべきかを選択できるような指標として利用いただければと思います。 <アジェンダ> - Photo Pickerとは？ - Photo Picker以外のメディアアクセス手法 - Photo Pickerと他のメディアアクセス手法の使い分け -（可能であれば）Photo Pickerってどうやって動いているの？	- 画像・動画等のメディアアクセスを利用するアプリの開発に携わっている方 - メディアアクセス周りのIntentの使い分けに悩んでいる方 - DroidKaigi 2021で発表された「メディアアクセス古今東西」をみた方	25 minutes	日本語 / Japanese	Android Framework	Yoshihiro Wada
急にチームが大きくなった話	新規開発のためにiOS 5名 Android1名というバランスで突如 あつめられたアベンジャーズ。最終的に半年でAndroid9名の大きなチームとなる。 大規模な開発になりそうな予感の中、1人でやっていくのはとても無理がある。 自身はリーダー経験もなければ、1人で大規模な新規開発をした経験もない中で、 さまざまなバックグラウンドを抱えたメンバーを集め、その中でどう開発を進めていったか。 自分自身何を反省し、何を学んだかお話したいです。	・大規模になりそうな開発をリーダー経験などがない方 ・Androidの人数がiOSよりもはるかに少ない立場の方 ・iOSからAndroidに移ろうと思っている方 ・エンジニア未経験の方で実務でアプリ開発したい方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	hiraok

Data Safety Section、App Privacy Details・ATTの基礎を知る	Google Playに登録する全てのアプリで、Data Safety Sectionによるアプリのプライバシーとセキュリティ方針の申告が必要になりました。「以前iOSでも似たような話があったな……」とApp StoreのApp Privacy Details・ATTを思い出した方も多かったのではないでしょうか。ストアからアプリを配信するには、これらのアプリのプライバシー情報の取り扱いについての質問・取り決めへの対応は必須です。今後もアプリ開発者はもちろん、アプリ開発者以外の開発関係者の理解が必要になるでしょう。
	
	Droidkaigiを機に、これまで対応してきたData Safety Section、App Privacy Details・ATTについて振り返りませんか。
	
	このセッションでは、Data Safety Section、Apple Privacy Details・ATTのそれぞれの公式ドキュメントの基礎的な内容と、弊社での取り組み事例を共有します。
	
	このセッションでする話
	
	- Data Safety SectionのGoogle Play公式ドキュメントの基本的な内容
	- App StoreのApp Privacy Details・ATTとの共通点、相違点
	- 弊社の事例
	  - 公式ドキュメントで推奨される対応
	  - プロダクト横断での情報共有
	  - 継続的に対応するための取り組み
	
	このセッションでしない話
	
	- 具体的な対応ケースの網羅
		- アプリエンジニアで、Data Safety SectionやApp StoreのApp Privacy Detail ・ATTを振り返りたい方
		- アプリエンジニアではないが、Data Safety SectionやApp StoreのApp Privacy Detail ・ATTをざっくり知っておきたい方
		- 他社のData Safety Sectionへの取り組みが気になる方
			25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	fusuma

実践 Glass EE2 向けアプリ開発	Google Glass のエンタープライズ向け最新モデル「Glass EE2」の販売が開始されて約3年が経過しました。 グローバルではいくつかの採用事例がありますが、国内ではまだマイナーな Android デバイスです。 一方で、このようなスマートグラスに対するユーザーの期待値は高く、近年は様々なメーカーからデバイスが販売されています。 本セッションでは、スマートグラスに期待されているユーザー体験を解説した後、Glass EE2 向けの開発環境やデザインガイドライン、アプリ開発例を紹介します。	- ニッチな Android デバイスが好きな方 ・スマートグラスに興味がある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	shanonim
マルチモジュールアプリの画面遷移処理実装	マルチモジュールで構成されたアプリでは、特に画面遷移処理が課題になりがちです。 最近では Navigation コンポーネントを使用した画面遷移解決が主流になりつつありますが、Navigation コンポーネントによるマルチモジュール間の画面遷移にはまだ制約があります。 クックパッドアプリでは Navigation コンポーネントの v1.0.0 が提供される以前からマルチモジュールでのアプリ開発に取り組んでおり、画面遷移処理を Navigation コンポーネントを使わずに自前で実装しています。長い歴史を持つクックパッドアプリは複雑な階層構造を持つ画面遷移とディープリンクを処理する必要があり、その高度な要求に対応するために画面遷移処理の実装を進化させてきました。 本セッションでは、去年から今年にかけて行われたクックパッドアプリの画面遷移処理のリファクタリングの経緯を元に、マルチモジュールアプリの画面遷移処理を効率的に実装するための設計について紹介します。	マルチモジュールアプリの画面遷移に困っている人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	こやまカニ大好き
継続的なライブラリアップデート手法とその効果	次のような、ライブラリに関する課題に遭遇したことはありませんか？ - バージョンが古いまま更新していないライブラリがある - AndroidXのような更新の早いライブラリのアップデートに追随できない - 更新があることに気づかない これらの課題は、ライブラリを継続的にアップデートする仕組みがあれば改善できます。 Renovateは、Mend社が提供するソフトウェアの依存関係を最新に保つアプリケーションです。 これにより、Androidプロジェクトの依存ライブラリを常に最新にできます。 本セッションでは、Renovateを使い継続的にライブラリをアップデートする手法と、実際に導入して得られた効果について紹介します。	ライブラリのアップデートに課題を持っている方 継続的にメンテナンスをしたい方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Masatoshi Kubode

Flutter におけるバックエンドと疎結合なソフトウェア・アーキテクチャ - 将来のインフラ移行に備える mBaaS との上手な付き合い方	近年、モバイルアプリのクロスプラットフォーム対応を進める目的で Flutter が導入されることが増えてきています。くわえて、とりわけ少数のチームで開発を進める場合、開発効率を高めるために Firebase や Amplify などの mBaaS が導入されることも少なくありません。そこで課題となるのが、中長期的な運用に足る拡張性やスケーラビリティと短期的な開発効率のバランスをどのように確保していくか、ということです。
	
	本セッションでは、mBaaS（2019年に採用）から 独自実装のサーバーサイドシステム（2022年7月運用開始）に移行した実例を元に、アプリ開発の初期段階から導入でき、将来のインフラ移行を念頭におけたアーキテクチャの実践例について解説を行います。
	
	本セッションをつうじて、聴衆はつぎのトピックについての知見を持ち帰ることができます。
	
	- Flutter でアプリケーションを開発する場合のアーキテクチャ例とそのポイント
	- Flutter の状態管理モデルの動向とプラクティス
	- 抽象化の実現に必要な Dart のクラス設計方法
	- Dart における Protocol Buffer + gRPC を用いたクライアントの実装例
	- mBaaS 採用における注意 点や pros / cons
	
	本セッションでは Flutter と Dart のコードを中心に取り扱いますが、Java, Kotlin を採用した Android アプリ開発においても、共通項を見出すことができる内容で構成されます。
		- 前提知識
		  - ソフトウェア・アーキテクチャ（MVC や オニオン、クリーンアーキテクチャ など）に関するごく初歩的な知識
		- 特に対象とする課題
		  - Flutter を採用しようか、採用してよいか迷っているかた
		  - Flutter でどのようなソフトウェアアーキテクチャを採用するとよいか考えているかた
		  - モバイルアプリ開発で mBaaS の採用を採用するか、採用する場合、しない場合の pros / cons について整理したいと考えているかた
			40 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	iktakahiro

Androidエンジニアのキャリアプラン〜中堅編〜	キャリアプラン、悩みますよね。 キャリアプランは個々人で様々なプランが存在し、これだ！という正解が無い世界です。 我々はこの登壇を通じてチャレンジの大切さを伝えたいと思います。 我々の所属するチームで ・別業界からAndroidエンジニアへ転向してきた ・Androidエンジニアかつエンジニアリングマネージャを行っている ・Androidエンジニアからプロジェクトマネージャへ転向する と様々なキャリアを歩めているメンバーがいます。 登壇者の2名のテックリードやプロジェクトマネージャーへの転向といったキャリアを中心に、以下のような観点を共有します。 ・登壇者やメンバーの経験から得られたキャリアプランの流れ ・各ロールごとにどういった職務を行っているのか ・今後のキャリアプラン ・現在のキャリアにどのようにしてなったのか、何故なったのか	・若手エンジニア ・キャリアプランを悩むエンジニア	40 minutes	日本語 / Japanese	その他 (Other)	thrasher
アプリを安全にリリースする	Androidアプリは、ストアを経由してユーザーにアップデートを配信します。アップデートは、ユーザーのコントロール下で行われます。アプリの提供側である我々は、アップデートをユーザーに届けることをコントロールできません。 もし、アプリが使えなくなる問題があるバージョンを配布してしまうと、ユーザーはアプリが使えない状態になり、プロダクトの大きな損失を生みます。 このため、Androidアプリのリリースには、高い品質が求められます。 弊社でもこのような問題をきっかけに、アプリを安全にリリースするフローの構築を行いました。 本セッションでは、弊社の実例をもとに、Androidアプリを安全にリリースする手法について紹介します。 CI/CD構築の例なども話す予定です。	アプリでインシデントを起こしたことがある方 品質を改善したい方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	Masatoshi Kubode
Androidアプリを8年運用する技術	Androidは発売から10年以上が経ち、規模の大きなアプリは5年10年と運用を続ける時代になってきました。市場の変化を捉えて新しい機能開発や改善に集中したいしたいところではありますが、Androidアプリは Webサービスと異なり新しいバージョンのみを考えて開発する訳にはいかず、過去バージョンを意識して開発していく必要があります。 この発表ではそういった長年開発を進めていく上でのポイントを整理し、新規開発に集中するための技術について紹介します。 - APIの互換性について。APIのバージョンアップをするときはどういうときか - 将来のパターン増加を見越してどうAPIから返ってくるレスポンスをパースするか - App Linksのパスをどう分けていくか - アプリbanをどう進めていくか	過去バージョンのことはさらっと考えて新規開発に集中したい方 Androidアプリを長く運用したい方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	phicdy
WearOSアプリ開発を始めてみよう！	Google I/O 2022で発表されたPixel Watchの発売楽しみですね。 Googleから久しぶりにスマートウォッチが発売されるので、WearOSアプリへの注目度が高まりそうですよね。 新しいWearOSではJetpack Composeでアプリを作成することも出来るようになったため、しばらくスマートウォッチのアプリ開発から離れてしまった人も再び興味を持つようになったのではないでしょうか？（私がそうでした。） そこで本セッションでは、一からWearOSアプリの開発を学びつつ新しいWearOSで可能になった機能のキャッチアップをしていこうと思います。 もしかしたら、DroidKaigi 2022当日に実機が手に入っていたら実機デバッグも出来たらいいなと考えています。 予定している内容 - ComposeでのWearOSアプリ開発の基本的な流れ - WearOSアプリのデザイン原則の基本 - Glanceでのタイル作成 - ウォッチフェイス・スタジオでノーコード・デザイン - WearOSアプリのデバッグ	・WearOSアプリ開発をこれから始めてみたい人 ・スマートウォッチのアプリ開発から離れていたけどまた興味が出てきた人	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	androhi
長期運用アプリのリファクタリングを考える	Android始まって以来の大きな転換期を迎えている今、リリースしてから年月が経過している長期運用アプリの中にはそもそも構造的に無理が生じているのも増えてきているのではないでしょうか？ Jetpack Composeが発表され、アプリアーキテクチャガイドでベストプラクティスが案内されていますが、試行錯誤を繰り返しながら運用しているアプリではなかなかついていくことに苦労することも多いです。運用中のアプリの開発を止めるわけにはいかずフルスクラッチという選択肢が取れないこともあるでしょう。 このセッションでは - ゴッドアクティビティが乱立しているアプリの分解方法 - 段階的なJetpack Compose化 - リファクタリングにおける効率的なAndroid Studioの使い方 など、アプリのアーキテクチャの方向性が明確化してきた中で長期運用しているアプリをリフレッシュする方法を考えます	長期運用しているアプリを抱えているエンジニア	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	shinmiy

Android Studio でコードを理解する技術	アプリ開発では、コードを書く時間よりも、読む時間の方が長い場合があります。 アプリが成長すればするほどコードは増え、新しくコードを書くことよりも、既存のコードを変更することの方が多くなっていきます。 そのため、ある日こんなことを思うのです。 ユーザーから問い合わせのあった不具合の原因がようやく特定できた。 あるメソッドの呼び出しをタイミングが適切ではなかったのだ。 タイミングを変更すれば不具合は解消されるが、そもそも、どうしてこのタイミングで呼び出す必要があったのだろう。一覧した限りでは必要性が理解できない。何か意図があったのだろうか。変更することで新たな不具合を発生させてしまうのではないだろうか。 どうしてこのような実装になっているのか理解しないと手が出せない。 本セッションでは、Androidアプリ開発で広く使われているAndroid Studioを用いて、既存のコードを理解するための以下のような内容を紹介する予定です。 ・コミット履歴の調べ方、歴史の追い方 ・コードを読みやすくする機能やプラグイン	・Android Studio で開発している人 ・コードを読むことに苦手意識がある人 ・今の実装に至った経緯を理解できるようになりたいと思っている人	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	yasuhiroki
既存画面にJetpack Composeを組み込む	既にAndroid Viewで実装し運用しているアプリにJetpack Composeを導入する方法には全部書き直す・新規画面からJetpackComposeで実装を行う・画面の一部に組み込むというようにいくつかあります。 全て書き直す方法は変更差分が大きくなりエ数も膨大になりやすいため難しく、新規画面から導入する方法も既存画面への機能追加や改修がメインとなるプロダクトフェーズではなかなか着手しにくい場合があります。 そういった状況の場合は既存画面の一部にJetpack Composeを組み込むという方法を取るケースがあります。 Android View利用時には複雑になりやすかった、複雑なアプリの状態に合わせて動的にViewを追加や削除、変更するということがJetpack Composeでは簡潔なコードで実現しやすくなるため、既存画面の一部を置き換えることで複雑さを低減することが期待できます。 また、画面の一部からという段階的な移行を行うことで移行を行うハードルを下げることもできます。 このセッションでは既存画面の一部にJetpackComposeを導入する方法を、Kotlinコードで導入する方法・カスタムビューとして導入する方法・Epoxyのような外部ライブラリで構築された画面に導入する方法の3軸を中心に解説を行います。 主に話すこと（予定） - Jetpack ComposeとAndroid Viewの相互運用性 - 既存アプリに導入する方法 　- 全て書き直す 　- 新規画面で導入する 　- 画面の一部に導入する - 画面の一部に導入する場合の導入手段 　- コード上でFragmentやActivityでの導入 　- カスタムビューとしての導入 　- Epoxyなどの外部ライブラリ利用画面での導入	既存アプリを運用しておりJetpack Compose導入に興味がある方 Jetpack ComposeとAndroid Viewの共存に興味がある方 Jetpack Compose in Android Viewに興味がある方	40 minutes	日本語 / Japanese	Jetpack Compose	mitohato14
アプリ内課金のエラーハンドリングを考える	Google Play Billing Libraryを利用しアプリ内課金用のViewを組み込み、アプリ内のコンテンツを購入・開放するという正常系の実装は公式ドキュメントや多くの記事・スライドで方法が解説されているため比較的スムーズに実装を行える場合が多いです。 ただ、アプリ内課金中に様々な原因で発生しうるエラーケースに対応した異常系の処理実装の記述は省略されているケースが多く、チームや実装者によって違う部分が多いと思います。 このセッションでは、アプリ内課金利用中に発生しうる異常系がどういったタイミングでどのように発生するか説明し、ユーザーと運営の両視点からどのように対応したほうがよさそうかをまとめ発表します。 主に話すこと（予定） - Google Play Billing Libraryでのアプリ内課金で発生しうるエラーの種類紹介 - 各エラーケースに応じたハンドリング手法の検討	アプリ内課金の導入を検討している方 アプリ内課金のエラーハンドリングを悩んだ過去がある方 エラーハンドリングに興味がある方	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	mitohato14
チーム開発におけるJetpack Compose設計戦略	弊プロダクト「スタディサプリENGLISH」Android版は現在7人チームでの開発を行っています。 すでにコードベースも30万行を超え、チームもコードも大規模化の一途をたどっています。 それゆえJetpack Composeを導入するに際してはチーム開発を前提としての設計を検討し、チームメンバー全員での合意形成や、迷わず実装できるようガイドラインの作成は避けられないタスクでした。 今回は弊チームがJetpack Composeを導入するにあたり何から着手したか、そして複数人で開発するにあたり策定したルールや気をつけたポイントなどをお話できればと思っています。 ポイントとしては「ComposeとViewModelの依存分離」や「UiStateを用いた統一感のあるデータの渡し方」、そして「デザインシステムの構築」や「Showkaseを用いたPreviewの活用方法」など、いわゆるPresentationレイヤーにフォーカスした設計戦略をお話する予定です。 また弊プロダクトにおけるAndroid ViewからのJetpack Compose移行をどのように計画立てているかも合わせてご紹介できればと思っています。 目次（予定） 1. Jetpack ComposeとViewModelの分離する 2. UiStateとUiModelを組み合わせてUI更新を行う 3. 整理されたPreviewのさらなる可能性を探る 4. デザインシステムをカスタムテーマとして活用する 5. Jetpack Composeへの移行計画の立て方	- 最近チームメンバーが増えてUI層の設計に悩んでる方 - Jetpack Composeを導入したけど無秩序になってしまって困っている方 - Android Viewで困ってないからとJetpack Compose導入に踏み切れていない人 - すでにAndroid Viewの既存資産が多くある中でJetpack Composeの導入をトライしている・しようとしている方	25 minutes	日本語 / Japanese	Jetpack Compose	kazakago
社内でのモバイルアクセシビリティ推進	担当アプリのアクセシビリティが気になっていても、 会社と言う環境で作るアプリはいろいろな人が関わっているため、新機能と比べてあまり魅力的でないものを簡単に取り込めないですね。 それにGoogleのガイドラインやコードでの対応方法を知っていても、Androidエンジニアの自分だけで出来る事の限界があります。 本当はデザイナー、PMやQAなど、アプリ関係者のみんなを巻き込む必要がありますのに、 実際いざとなると、どこからどうすれば良いか分からない。。。 Webアクセシビリティのガイドラインを読んでもピンと来ない。。。 デザインとかiOSなど自分の担当領域以外のことは詳しくなくて質問に答えれない。。。 心が折れてしまいそうですね。 せっかくアクセシビリティ強化を考えている方がそんな壁にぶつからないように、本セッションでは、現在社内でアクセシビリティを推進している私の通ってきた道と、集めてきた情報を話します。 絶対に聞かれる「なぜアクセシビリティ対応をするか」の答え方、社内勉強会の内容、対応範囲の決め方や具体的な取り込み方など、必要な参考情報を共有します。 ご自身の担当アプリのアクセシビリティ改善がうまく行きますように！	アクセシビリティにちょっと興味があるけど、仕事にどう取り込めるか分からないAndroidエンジニアの方。 担当アプリのアクセシビリティを改善するために、推進方法の参考が欲しいAndroidエンジニアの方。	40 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Tiphaine (ティフェン)
Android業界をぶち上げるための最適解はなにか	私たちはAndroid初心者・中級者向けの情報サイト「Android Cafe」を運営する22卒,23卒のAndroidエンジニアです。 今回はこのAndroid業界をぶち上げるために何をすればいいのかについて私たちなりにまとめました。 Androidエンジニアの母数は日本では少なく、実際にAndroidのエンジニアをしていて、人数が少ない&独自として聞くことが少ないことなどから「Androidエンジニアという職種について認知されていない」というのを感じることがあります。 認知はされていても、Flutterなどのクロスプラットフォームでの開発なども可能であること、iosのシェア率が圧倒的に高いことなどから人口がそちらに流れていきます。そんなや、若手の私たちがこのAndroid業界を引っ張る必要があるのではないかと感じ、若手で集まって議論しました。その議論で繰り広げられた内容についてこのセッションでは述べます。 このセッションを聞いて、多くの人がこの業界に興味を持ってくれたり、少しでもこの業界を盛り上げようと思っていただけると嬉しいです。	・Android業界を盛り上げたい人 ・Android開発をはじめて間もない人 ・今後Androidエンジニアとして活躍していきたい人 ・学生	40 minutes	日本語 / Japanese	その他 (Other)	ぶる, すいみー, みっちゃん
Androidを選択した学生が爆速成長するまでの軌跡	22卒の新社会人のみっちゃん、23卒の学生のぶる、すいみーです。 私たちは、Androidエンジニアとして実際に実務をこなし、これまでAndroidエンジニアとしてインターンシップや、「Android Cafe」と呼ばれるAndroidエンジニア初心者・中級者用情報サイトの運営などの取り組みをやってきました。 学生や、社会人でキャリアを見直したいという人で以下のような課題を持つ人も少なくないと思います。 ・Android開発がしたいけど何を勉強すればいいのかわからない ・そもそもAndroid開発は魅力的なのか ・Androidエンジニアインターンシップは実務経験必須だらけで辛い ・キャリアの見直しで爆速成長したいけど方法がわからない そこで、私を含む最前線で活躍する若手エンジニアたちが、なぜAndroid開発を選び、学生の身分でどのようにして勉強し、爆速成長したのかについてお話していきます！ この業界や成長に不安を持っている人たちにとって、少しでも前進する希望になってくれると嬉しいです！少しでも多くの人がこのAndroid業界を選んで、今後活躍してくることを願っています！ご参加お待ちしております！	・Android開発初心者 ・進む道に迷っている学生 ・エンジニアインターンシップに参加したい学生 ・キャリアを見直したい方 ・Android開発を選択したが、成長を感じれていない方 ・Androidエンジニアとして活躍していきたい方 ・Android開発の勉強方法を知りたい方 ・ワイワイしたい方	40 minutes	日本語 / Japanese	その他 (Other)	ぶる, みっちゃん, すいみー
Jetpack Composeでインタラクティブなアニメーションを実装しよう	Jetpack Composeではユーザーの操作に伴ったインタラクティブなアニメーションを簡単に実装することが可能です。例えば、HorizontalPagerのスクロールに応じてコンテンツのサイズを拡縮したり、Column/LazyColumnのスクロールに応じてコンテンツのImageをパラックスさせたりなどを簡単に実装できます。 従来のViewの世界ではこういったアニメーションの実装方法はViewごとに方法が違ったり、そもそも方法が存在しないためViewを継承してカスタマイズしたりする必要がありましたが、Jetpack ComposeではgraphicsLayerというAPIを駆使することである程度統一的な手法でこういったアニメーションを実装できます。 このセッションでは、graphicsLayerをつかったインタラクティブなアニメーションの実装方法を実例を伴って紹介します。	Jetpack Composeでのインタラクティブなアニメーションの実装に興味のある方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Moyuru
Compose の Preview どうする問題	皆さん Jetpack Compose 使っていますか？Preview 便利ですよね。Preview は便利ですが、どう Composable 開数を渡したらよいか迷いませんか？私は迷っています。 ステートレスな Composable のみを Preview する？途中に100%ステートレスではない Composable が混ざっている Screen 全体の Preview をしたい場合は？DroidKaigi 2021アプリのように FakeViewModel のようなものを作り Preview 用に渡す？ 本セッションでは、（恐らく）公式で推奨されている作り方を含め様々な Preview 可能な方法を調査したうえで、それぞれに関して考察します。	Jetpack Compose の基礎知識がある方 Jetpack Compose の Preview に悩んでいる方	25 minutes	日本語 / Japanese	Jetpack Compose	Pon

システムUIを考慮した堅牢な画面の構築 in 2022	DroidKaigi 2018において「ウィンドウサイズの変更に強い堅牢な画面の構築」というテーマで発表をさせていただきました。このセッションの前半で触れたのは "ステータスバーなどのシステムUIの高さをいかにして考慮すべきか" 。WindowInsets APIやレイアウトファイルにおけるfitsSystemWindows属性の振る舞いについて解説しました。 当時の最新のAndroidバージョンは8.1 Oreo。以降5回のメジャーアップデートではシステムUI周りにも多くの変更が行われています。"Edge-to-edge" と呼ばれるフルスクリーンでの描画が推奨される中で、システムUIを正しく考慮する重要性も高まってきました。 本セッションではここ数年のシステムUI関連のアップデートをキャッチアップしつつ、システムUIを正しく考慮するためのTips / 知見を共有させていただく予定です。 《内容（予定）》 ・直近のシステムUI関連のアップデートのキャッチアップ ・Edge-to-edgeのレイアウト構築 / Synchronized IME transitionsなどの実践 ・システムUIの配置や配色を考慮する上での注意すべき点 ・Jetpack ComposeにおいてWindowInsetsを考慮する手法	・直近のシステムUI関連のアップデートについてキャッチアップしたい方 ・多様なデバイスやウィンドウサイズにおいても表示の崩れない画面の構築に興味のある方 ・Jetpack ComposeにおけるWindowInsetsの考慮について興味のある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	nakamuuu
All your Compose @Previews to screenshot tests without instrumentation	Screenshot tests are great way to test your view layer and catch many regressions. Jetpack Compose already has a handy @Preview feature for the IDE - and you can automatically make then into screenshot tests. With the help of Paparazzi library, you can capture screenshots without device or emulator, which is cumbersome on CI. During this session I will share our screenshot testing experience from a large enterprise app including GitHub Actions CI integration.	The talk is accessible to everyone, but interest in Android testing is welcomed.	25 minutes	English	保守・運用・テスト (Maintenance, Operations, and Testing)	David Vávra
Android歴0でもできるCameraXへの移行方法	CameraXを利用するとライフサイクルに沿ってカメラを起動可能であり、画像の向きや端末の向きなどを考慮してくれたりとカメラ周りの実装を簡単に行うことが可能である。またGoogleがさまざまな端末やOSバージョンでテスト全しており安心して利用できる。 今回弊社では特定の端末でカメラが起動しない問題を解消するため、既存のカメラライブラリ（fotoapparat）からCameraXへのカメラライブラリの移行作業を行った。私はこのライブラリ移行作業がAndroidエンジニアとして初めての仕事であったため、CameraXの学習からスタートすることとなった。私のように既存のカメラライブラリからCameraXへ移行したいがCameraXを触ったことがないと考えている方に向けて、CameraXの学習方法を紹介する。 またAndroid端末は数多く存在しすべての端末でテストすることはできず想定外の挙動をする可能性がある。その例として端末の充電が10%以下の場合にカメラを起動できない端末、インカメラが存在しない端末、フラッシュ機能が存在しない端末がある。このような端末において場合によってはクラッシュするケースも存在するため、CameraXでデバイスの状態をチェックする方法、例外処理についてご紹介する。 そのようなケースに備えても想定しきれない問題が発生する可能性、また単に実装の不便でライブラリの移行に失敗する可能性も存在する。そのためにRemoteConfigを利用した切り戻し機構のライブラリ移行手順を紹介する。 このセッションではCameraXの学習方法、実装例、弊社で行ったライブラリ移行手順をお伝えし、聴講者の方に安全かつ確実なライブラリ移行を実現できる安心感をお伝えします。	CameraXの学習方法がわからない方 CameraXが何か知りたい方/移行するメリットを知りたい方 Cameraライブラリ移行例を知りたい方	25 minutes	日本語 / Japanese	ハードウェア (Hardware)	Tomonori Matsukawa
マンガアプリのメモリ改善とメモリ解析方法	1. Lineマンガで8年運用したAndroidアプリのレガシーの問題としてOOMのメモリクラッシュ問題を真剣に向き合って解決した事例を紹介します。2. AndroidのVMの発展過程の紹介とそのメモリ管理方法とGCの管理方法の変化を辿る。3.基本的なGC（Garbage collection）の考え方からAndroidのメモリ解析のための基礎知識を伝える。4. 具体的なメモリ解析の方法を事例で紹介する。5. メモリリークを防ぐため、メモリを改善するための注意点と実際の例を紹介する。6.Google I/O 2022のLarge Screen重視でさらに重要になったConfiguration changeのメモリ対策	JVMメモリ管理方法、GC(Garbage collection)に聞心がある方、Androidメモリ、パフォーマンスを改善したい方、 マンガが好きな方。	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	sai.choko

Dive into Modifier	# 概要
	
	Modifier は Jetpack Compose の UI 要素を装飾・拡張したり外部の振る舞いを設定するために使われる Kotlin のオブジェクトです.
	
	Modifier を使うことによって Composable に対してサイズ, レイアウト, 動作, 外観を変更したり, ユーザー補助ラベルなどの情報を追加したりといった装飾を行うことができます.
	
	しかし, Modifier は直感的に記述をすることができるので何となくで使っている人も多いのではないでしょうか?
	
	そこで本セッションでは, 実装と検証を通して Modifier の深堀とプロジェクトにおける Modifier の良い使い方について紹介します.
	
	具体的には最初に, Modifier の概要と定義・実装について確認していきます.
	
	次に Modifier の特徴についてpadding や size 等の具体的な修飾子を題材にし, 内部の実装の確認と検証を通して深堀をしていきます.
	
	最後に API Guidelines for Jetpack Compose やプロジェクト開発における知見をもとに Modifier の実践的な使い方について紹介していきます.
	
	# 内容
	
	- Modifierの概要と定義・実装
	- Modifierの特徴
	  - Scope
	  - Order
	  - Chain-Combine
	  - Repeatability
	- Modifier実践
	  - プロジェクト開発において意識すべきこと
	  - Custom Modifier
		- Jetpack Compose を触っている・興味がある人
		- Jetpack Compose の Modifier に詳しくなりたい人
		- Jetpack Compose の内部実装に興味がある人
			25 minutes	日本語 / Japanese	Jetpack Compose	403

Themed app icons	Android 13 から Themed app icons という機能が追加されます。この機能によって、ユーザーが選んだ壁紙やその他のテーマで色付けされたアプリアイコンを表示できるようになります。 Android 12 で追加された Dynamic colors とあわせて対応することで、Android 端末全体のテーマをより統一できるため、今後、少しずつユーザーから要望が上がってくるのではないでしょうか。 このセッションでは、Themed app icons の仕組みや対応する際の注意点などについて詳しくお話しします。	- Themed app icons のことを詳しく知りたい方 - Themed app icons の対応を検討している方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	blendthink
Jetpack ComposeとMaterial Design 3でTwitterクライアントアプリを作る！	本セッションでは、Jetpack ComposeとMaterial Design 3を使用してTwitter（API v2）クライアントアプリを作る方法について、具体的な実装方法を解説します。 タイムライン表示、写真表示、...、認証コールバックのハンドリングまで、Twitterクライアントでお馴染みの、いくつかの機能の実装例を紹介する予定です。	- 初心者向け - Jetpack Compose に興味がある人	25 minutes	日本語 / Japanese	Jetpack Compose	adakoda

Jetpack Compose + GraphQLによるGuide to app architectureの実践	Googleの推奨するアプリアーキテクチャのガイド（Guide to app architecture、以下ガイド）は、昨年内容が大きく刷新され、UI、ドメイン、データとレイヤ毎のより詳細な説明が追加されました。具体的な指定が増えて、すべきことがより明確になった一方、どこまでガイドに従うか、自分たちの要件に適合しているか、といった落とし所の判断は、いざ実践していくと難しいところがあります。
	
	そこで、本セッションでは、Jetpack Compose + GraphQLによる新規アプリ開発を題材に、実際にガイドに沿った実装をしていく上で悩むであろう部分や、なぜガイドがこうしているのか、そのうえで自分たちの要件にどう合わせていくか、といった部分を、実例をもって説明します。皆さんがガイドを自分のアプリに活用していく上での、考え方のヒントになれば思います。
	
	大まかなトピックとしては、次を予定しています。
	
	- UIレイヤ
	  - Composeで作ったときに何が起こったか
	  - ステートホルダー、どう作る？
	  - イベントの再整理：UIロジックとビジネスロジック
	- ドメインレイヤ
	  - Compose + GraphQLで作ったときに何が起こったか
	  - ドメインレイヤの必要性を考える
	  - UseCaseからUseCaseへの依存など、依存関係の設計をどうするか
	- データレイヤ
	  - GraphQLで作ったときに何が起こったか
	  - 理想と現実、抽象化と開発効率
	  - モデルをどう作るか
	- 関連
	  - マルチモジュールとアーキテクチャ
	  - テストとUIライティング、DI
		- アプリアーキテクチャにお悩みの方
		- Google推奨のアプリアーキテクチャの導入事例を聞きたい方
			40 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	haru067

AndroidManifest.xmlはその後どうなっているのか？	Android開発している必ず編集するであろうAndroidManifest.xmlですが、このxmlファイルがその後どうなっているかご存知でしょうか？ アプリに組み込まれて必要に応じて適宜参照されるのかなというイメージはあるかもしれませんが、実際に追ってみたことはありますか？ このセッションでは、AndroidManifest.xmlがどうアプリに組み込まれ、設定した値がどのように使用されているのかを解説します。	- Android開発をしている人 - AndroidManifest.xmlがどう使われているか知らない人	25 minutes	日本語 / Japanese	Android Framework	takasy
ログと戦う	ユーザーの行動ログはアプリ改善のために必要な情報です。 しかしその実装には大きな苦労を伴います。ログのために綺麗にビジネスロジックが歪んだコードへログ送信コードが追記されたり、ログ実装のために構造自体を変更せざるを得なかったり、そもそもどうやってこのログを取ればいいのかわからない、といった問題に直面します。ユーザーに良い体験を届ける第一目標を掲げつつ、エンジニアはログ送信機能を実装していかなければなりません。 本セッションではどのようにログ送信を実装していけば、本質のコードに対して少ない影響で実装していけるか、またJetpackComposeにおける表示ログの実装方法などを実際の開発事例に基づいてお話致します。	- アプリ内の行動ログがうまく取れずに泣く泣く実装を断っている方 - なるべく既存の実装を綺麗に保ちつつログを送りたい方 - ログの実装に疲れた方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	kuromame

Androidエンジニア0人の状態から始めるAndroidアプリ開発	私の所属する部署では事業成長のためにAndroidアプリの機能開発をしたい要望があるものの、Androidエンジニアがチームにいない状態でした。Androidエンジニアが0人の状態からチーム内でAndroidアプリが開発できる状態になるまで、どういった意思決定や行動があったのかをお伝えします。 最終的に元iOSエンジニアと元サーバサイドエンジニアの2人でアプリを開発することになったのですが、その際のAndroidアプリ開発のキャッチアップや他プラットフォームでの知識を活かした技術選定についてもお伝えします。 ■ 予定している内容 - Androidエンジニア0人からアプリを開発するために必要だった意思決定や行動 - Androidアプリ開発のキャッチアップについてと社内で得られたサポート - 他プラットフォームでの知識を活かした技術選定 - SwiftUIの知見を活かしてJetpack Composeでアプリを開発する - 社内の既存リソースを活用する	既に他プラットフォームでの開発経験があり、Androidアプリ開発に挑戦したい方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	natmark, Hyoga
ヘルスケアデータの連携機能、いま選ぶなら？	ウェアラブルデバイスや生体情報の計測機器の発展に伴い、ヘルスケアアプリにおけるデータ連携機能はあって当然のものになりつつあります。 そんな中、Google I/O 2022では、Health Connectが発表されました。そして今までAndroidアプリのヘルスケアデータ連携における重要な選択肢だったGoogle Fit Android APIが非推奨となりました。 Androidアプリにおけるヘルスケアデータ連携にはいくつか選択肢がありますが、Google Fit（Android/REST）やHealth Connectなどをプロダクトに取り入れるために知っておきたいことをヘルスケアアプリのAndroidエンジニアが紹介します。	ヘルスケアモバイルアプリケーションの新規事業や企画の担当者、およびPMやAndroidエンジニア、個人開発者の方々	25 minutes	日本語 / Japanese	Jetpack	Seigo Ohzono
動画プレイヤーにChromecast対応を入れてみよう	動画プレイヤー機能を持つアプリに関わっていると、Chromecastデバイスへの対応が要望としてよく上がってきますが、 逆に関わってない場合、Chromecast機能の開発自体に馴染みがないのではないでしょうか。 また初めて取り掛かる場合、Cast SDKの導入方法から、セッション状態の監視、UI調整方法など考慮すべき点が多いので、苦労する点が多いと思います。 本セッションでは - Chromecastデバイスへの接続から映像の再生まで - キャスト状態の監視方法 - ExoPlayerとの連携 - Senderコントローラー等のUIをカスタムする方法 - 開発する上とで苦労する点 を紹介し、Chromecast対応へのハードルを少しでも下げられたらと思います。	Chromecastデバイス対応に少しでも興味がある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	takathemax
今一度スプラッシュスクリーンに取り組む	Android12では新しくシステムデフォルトのスプラッシュ画面が常に適応されるよう変更され、ランチャーアイコンがスプラッシュスクリーンに表示されるようになっています。 スプラッシュスクリーンはブランドアイデンティティを確立させ、待ち時間の体験を向上させるためのエンゲージメントとして最適な解です。 本セッションでは、Android12と11以前との共存方法を考えながら、画面要件の対応方法もご紹介します。 ■アジェンダ（予定） ・Android12と11以前の違い ・互換性ライブラリの使用 ・新スプラッシュ画面要件 ・スプラッシュ画面のカスタマイズ	・スプラッシュスクリーンでどんなことができるのか知りたい方 ・これからスプラッシュスクリーンを取り組む方 ・ブランディングでスプラッシュスクリーンを活用している方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	shuhei yamamoto
Android(AOSP)でウェアラブルIoTデバイスを作ってみた	私たちはウェアラブルIoTデバイスを開発するにあたり、OSとしてAndroid Open Source Project(AOSP)を採用しました。 本発表ではAOSPを採用するに至った経緯や、ウェアラブルIoTデバイスとして開発するとで苦労した点・工夫した点について紹介させていただきます。 具体的には以下のような内容について紹介します。 - デバイス内のアプリをマイクロサービス的に設計・実現 - Play Store／開発者サービスを使わずに運用するためのアレコレ - 画面がない！困ったことと対処 本デバイスは現時点で数千台が世に出ており、特に画面がないことについては現在もデバイスを運用しながら良い方法を模索し続けています。 デバイスを要サービスとして運用する上での試行錯誤についてもご紹介させていただきます。	- 組み込み(embed) Android技術者 - 独自デバイス上でAndroid(AOSP)を動かすことに興味のある方 - IoTデバイスを使ったサービス開発・運用に興味がある方	25 minutes	日本語 / Japanese	Android Framework	Yoshinori Mukai
今一度「デプロイ」を考えてみる	「なんとなく」でデプロイ環境を運用していて思いが付け落とし穴にはまってしまったことはありませんか？ 本セッションでは、デプロイに関する理解を深め、リスクマネジメントや開発者/利用者双方の利便性向上につながる知識を共有します。	・今までなんとなくデプロイ環境を構築/利用してきた方 ・これからデプロイ環境を導入する方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	TakO
開発してみてわかった便利なコンポーネント	Android開発を始めて一年、 ViewModelやLiveData、Navigationなど、 使ってみて個人的に感動した便利なコンポーネントについて、 実際の使用例を交えながら、ご紹介したいと思います。	Android開発初級者	25 minutes	日本語 / Japanese	Jetpack	masaki
雰囲気で利用しないためのNavigation with Compose入門	ComposeにおけるNavigationは非常に強力で使いやすく、ある程度雰囲気で実装することが可能です。しかし雰囲気で実装している上実際にナビゲーション上の問題に直面した時（例えば戻ってくると画面が真っ白になっていたり、想定と違う画面にいたり）の対応が困難です。 本セッションではNavControllerのvisibleEntriesやcurrentDestinationを追いつつ、Compose間や既存Activityへの画面遷移でどのように動いているかについてお話しつつ、登場するコンポーネントについて解説いたします。また、私が雰囲気で実装していた時に直面した問題やアンチパターンについてもお話しできたらと思います。	- Jetpack ComposeのNavigationをこれから触る予定の方 - NavGraphやBackStackEntryについてあまり詳しくない方 - 既存画面の一部だけJetpack Composeに置き換えたい方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	kuromame
In-App Review, In-App Updatesの活用	Androidアプリはたくさんのアップデートやレビューが行われています。これらはPlaystoreを介して操作が行われることが多く、アップデートなら自動アップデートあるいは自ら手動でアップデート、レビューなら手動で対象のアプリを検索して評価が行われています。 これらの課題として、ユーザーにいち早く新機能を提供したり、不具合の修正などのアップデートをしたいときにユーザーがアップデートに気づかなかったり、アプリから一時的に離れてしまったりする不便さが考えられます。 本セッションでは In-App Review, In-App Updates をプロダクトで実装した際の課題や改善、ノウハウについて紹介したいと思います。	Androidアプリ内でレビューやアップデートを利用・検討している方	25 minutes	日本語 / Japanese	Kotlin	Yuma Kanno
Material Design における Typography のこれまでとこれから	見過ごされがちですが、Typography はユーザーインタフェースの大部分を占める要素であり、デザインシステムにおいても重要な要素のひとつです。 昨年、Android 12 のリリースとともに Material Design 3 が公開され、ドキュメントの内容が刷新されました。 本セッションでは、中でも Typography に焦点をあてて、どのように変化したのかを解説します。 新しくなった Typography でできることを知っていただき、実際にアプリに取り入れて活用するにあたっての要点を整理します。 ・従来の Material Design 2 における Typography ・より規則的でシンプルになった Type scale ・デザイントークンによる Typography の管理 ・Typography としての Material Symbols の活用 ・日本語と Typography ・アプリに適用するにあたっての要点	・Material Design 3 の導入を考えている方 ・Typography を扱う上での要点を知りたい方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Ryota Konno
なんとなくのContext使いから脱却する	Android開発をしているとContextに遭遇することが多いです。 具体的にはダイアログの生成やデータベースのセットアップなどです。 それほど多用しているにも関わらずそれがどのような役割を果たしているのかをよく理解せずに使っていることが多いかと思います。 基本的にはメソッドの型に従って適当なContextをセットするだけで問題なく想定した挙動を確認できるでしょう。 しかし、なりふり構わずContextを使用しているとメモリリークを引き起こす場合があります。 Contextをアプリ上における「運び屋」であるという定義のもと、Contextの実態について解説します。 具体的には下記について話します。 - Contextとは一体何か？ - なぜContextを渡す必要があるのか？ - なぜリソースのアクセスにContextが必要なのか？ - activityContextとapplicationContextの違い - ViewModelでContextを持たせるとメモリリークを引き起こす理由 - composeにおけるCompositionLocalの使い方とそのデメリット - LookAhead下moveableContentOfを用いた明示的な値渡し composableでは、APIとして提供されているComposition Localを用います。 しかしこれは、暗黙的な受け渡しで便利な反面、その便利さゆえに保守性が落ちることが考えられます。 そこで、compose-1.3.0-alpha01よりリリースされたLoockedAheadやmoveableContentOfで明示的な値渡しをするというソリューションについても解説します。	Contextをなんとなく使っている方 Contextの正体を知りたい方 ComposeにおけるConextの扱いを知りたい方	25 minutes	日本語 / Japanese	Android Framework	すいみー
Google Play Media Experience programの技術的対応	Google Media Experience programには大まかにLarge screen対応とEntertainment space(tablet向け)のものがありましてこの二つを条件にGoogle側から手数料を安く受けられる。 このセッションでは 1. Google Play Media Experience programに合格するための最小限のノウハウ。 2. Large screen対応のためにやったことと落とし穴、解決策、 3. Entertainment spaceの背景と技術的対応方法 4. Google IO 2022で見えたLargeScreen向けの本気度 をメインで話ができればと思います。	比較的規模があるアプリ開発者と企画の方、よりよいGoogle playの手数料条件を受けたい会社の方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	sai.choko

簡単な家計簿アプリを作りながら学ぶ、AndroidのUnitテスト入門	仕様変更や機能追加をおこなった際、わざわざアプリをビルドして変更を確認してないでしょうか。 少しの変更を確認したいだけなのに面倒だなと感じたことがあるかと思います。 そんな時にUnitテストを記載すると変更の確認が非常に楽になります。 また、Unitテストを書きたいと思いながらも、どう書いていいかわからず敬遠してしまっていたり、 テストを書いてはいるものの意志を感じられない、運用しているうちにテストがメンテされなくなっている 最後にはUnitテストを書かなくなった経験がある人もいるかと思います。 この発表ではUnitテストへの入門と実際に運用していく方法を理解することを目的とし、 簡単な家計簿アプリを作成しながら実際にViewModelにUnitテストを作成し具体的に以下のことを学習していきます。 - 何故Unitテストが必要なのか - Unitテストを導入した上で得られる恩恵について - 良いUnitテスト、悪いUnitテストは実際どんなものなのか - 良いUnitテストの書き方 - 良いUnitテストを書く上でのクラス/メソッド設計の理解 - Unitテストを運用していく上で注意すること - 運用していく中で起きるミスとその対策	- Android開発初学者 - Unitテストを書いたことがない方 - Unitテストを書いているがなんとなく書いている方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Kosuke Sasaki
Paging3への移行で起きた問題とTips	大きなデータを小さなデータに分割して必要な分だけ読み込みを行うページング処理を簡単に実装できる公式ライブラリにPagingライブラリがあります。 PayPayフリマのAndoridアプリでは、おすすめの商品などの大きなデータをユーザーに遅延などでストレスを与えないように見せるため、 Paging2を使用しています。 2021年5月5日にPaging3がリリースされてから、Paging2がDepreatedになりました。 そのため、Paging3にマイグレーションする必要があり、PayPayフリマでも移行を進めています。 移行するうえで起きた ・容易に起こりえるがAPIの無限呼び出しが発生する致命的なコード ・移行する上で問題になったDatabase周りの独自実装 ・StateAdapterのパフォーマンス改善 という実際で移行する上で得たPaging3に関する知見である上記3点を紹介したいと思います。	・Pagingの基礎知識がある方 ・業務や個人開発でPaging3への移行をする人	25 minutes	日本語 / Japanese	Jetpack	ふぇり
俺の考えた最強のボトムナビゲーションの作り方	ボトムナビゲーションはモバイルアプリにおける普遍的なデザインのパターンです。 AndroidではMaterial ComponentsライブラリのBottomNavigationViewが提供されていますが、かって、これはMultiple Backstackをサポートしておらず、「iOSのような」ボトムナビゲーションを実現しようとする場合、自分で実装するしかありませんでした。 昨年Fragmentがバージョン1.4.0で作り直され、Multiple Backstackがサポートされるようになり、ライブラリのアップデートをするだけで、BottomNavigationViewでもMultipleBackstackが実現できるようになりました。 喜び勇んでライブラリをアップデートして試したところ、その ままでは想定通りの動きをしない場面があることがわかった。 なんなら、これまでどおりのほうがまだマシとさえいえる場面まであった。 時は寄しも新規アプリ開発中であった。 未来に期待して古いバージョンを使用するか、アプリの仕様を変えるかを迫られた私だったが、ここで第三の秘策、自作することを選択した。 FragmentManagerの新規APIを使用すれば、Multiple Backstackをサポートしながら、理想的な動きをするナビゲーションを作ることが容易にできる。 せっかくなので、夢いっぱい詰め込むことにした。 このセッションでは、理想的なナビゲーションのありようについて考察したあとで、その具体的な実装方法と、ディープリンクとの統合、運用時の注意点について解説します。	・マルチモジュールでアプリ開発をしている人 ・画面遷移にこだわりのある人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Hiroyuki Mori
やってみようMaven！	みなさんはAndroidのライブラリやSDKを作ったことはありますか？ライブラリの開発には、アプリ開発とはまた違った難しや楽しさがありますよね。 公開の方法もアプリとは異なり、aarファイルをmavenを利用して管理・配布するのが一般的かと思います。本セッションでは、この際によく利用されるであろうGradleのmaven-publishプラグインに焦点を当てます。プラグインの基本的な使い方を中心に、 - mavenの基礎知識 - maven-publishの使い方 - maven-publishで生成されるpom.xmlについて 　- Gradle内の記述がpomにどう反映されるのかを見ていきます - ローカルリポジトリの使い方と活用方法 などの情報・知見の共有を予定しています。 また、ライブラリを利用する側のお話として、repository filterの定義方法についても触れる予定です。 Mavenに触れたことがない方や、maven-publishプラグインをなんとなく使っている方、一緒にMavenの世界を覗いてみませんか？	- Mavenやmaven-publishプラグインに初めて触れる人 - Androidライブラリを開発・公開しようと考えている人 - 非推奨となったmavenプラグインからの移行を検討している人	25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	n-seki
5chマイクを使ってビームフォーミングできるようにAudio Frameworkを魔改造する	ビームフォーミングとは複数のマイクを使って、特定方向の音声を強調しつつそれ以外の音声を抑制した音を取り出すことを実現する技術であり、モバイルPCやWebカメラ、ヘッドセットなどにも使われている技術です。 ビームフォーミング処理は、専用ハードウェアチップで実装されているケースが一般的ですが、ソフトウェアで実装することも可能です。 私たちは、Androidベースのデバイスでもビームフォーミングを扱えるようにするべく、5chマイク(5つのマイク)を扱えるようにしつつビームフォーミングライブラリを組み込むために、Android OS のAudio Framework と格闘したのでその記録を紹介します。 本発表では、ビームフォーミングの仕組みやAudio Frameworkの構成などにも触れつつ、実際にどのような作業によってビームフォーミングを実現したのかを説明します。 具体的には以下のような作業を行う必要があります。 - Audio Framework にビームフォーミングのライブラリを組み込む。 - 5chマイクからの入力を扱えるようにする - アプリからは1chのデータのみが入力されているように見えるようにする - 5chマイク以外からの入力（イヤホンマイクやBluetoothヘッドセット）との切り替えをいい感じに実装する 音声処理に興味がある方だけでなく、普段あまり触れることの少ないAndroid Frameworkを直接改造するという試みに関心のある方にも興味深い内容になると思います。	- 組み込み（embed）Android技術者 - 独自デバイス上でAndroid(AOSP)を動かすことに興味のある方 - 特にマイク・音声処理周りに興味のある・仕事にしている方	25 minutes	日本語 / Japanese	Android Framework	Akira Kimura
実例から学ぶJetpack Composeのパフォーマンス改善	Jetpack ComposeはAndroid Viewシステムの教訓を経て、多くのケースで優れたパフォーマンスを発揮します。一方で、いつの間にかパフォーマンスが低下している経験をした人も多いのではないでしょうか？特にアニメーションの表現や、スクロールやドラッグの操作によってUIを切り替える際に大 問題が発生します。これらは、必要以上の recomposition で高速に更新処理が行われることに起因します。他にも、Lazyリストを使う際、RecyclerViewとJetpack Composeを組み合わせる際等にも注意すべき点があります。 より良いパフォーマンスを得るには、Jetpack ComposeのUI更新の仕組みを理解し、状況に応じて対処していく必要があります。このセッションでは実例をもとに、パフォーマンス低下の原因、計測方法、改善方法について解説を行います。	・Jetpack Composeを既に利用している人 ・Jetpack Composeにおいてパフォーマンスに困っている人	25 minutes	日本語 / Japanese	Jetpack Compose	Mori Atsushi
Jetpack ComposeでWindow Insetsと戦う	モダンなAndroidアプリを開発する上で、Window Insetsを正しく扱うことは、もはや必須のスキルと言えるでしょう。Window Insetsを考慮することで、Navigation BarやStatus Barの裏にコンテンツを描画したり、ソフトウェアキーボードの表示をより細かく制御できるようになります。一方で、Androidデバイスの多さや、OSによるAPI差分から、Window Insetsは非常に難解です。 Jetpack Composeはv1.2.0からWindow Insetsをサポートするようになりました。Jetpack Composeを使うことで、Android Viewよりも簡単にWindow Insetsを制御することができます。 このセッションでは、Jetpack Composeを使ったWindow Insetsの制御方法について詳しく解説します。	・Jetpack Composeを既に利用している人 ・Window Insetsについて詳しく知りたい人 ・モダンなAndroidアプリのUIを実現したい人	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Mori Atsushi
一歩先のAndroid Clean Architecture 〜いるレイヤー、いらないレイヤー〜	Androidのアーキテクチャを考えるときに、UIレイヤーにViewとViweModel、データレイヤーにAPI/DBとRepository、後は場合によってドメインレイヤー（UseCase）があれば十分と考えていませんか？確かにこれは推奨されているアーキテクチャであり、多くの場合で理想的に機能するでしょう。 しかし、このアーキテクチャはアプリの内容も開発チームの構成も考慮していません。単純なTODOアプリといくつもの機能を持ったSNSアプリを同じアーキテクチャで作成できるでしょうか？本当にそのレイヤーは必要ですか？十分に足りてますか？ 私は、優れたアーキテクチャは対象とするアプリケーションを正しく表現していると考えています。必要に応じて、さらに様々なレイヤーがあり、それらの境界が存在します。例えばチャットと電話の機能を提供するような、複数の機能が存在するアプリケーションでは、各機能をできるだけ分離させるほうが望ましいでしょう。また、複数の国でサービスを提供しているアプリケーションにはどこかに言語のレイヤーが存在するはずで、Android TVやiOS(KMM)のように複数プラットフォームに配信する場合はその境界を考えなければいけません。また、その境界をどのように明示するのか、packageで区切るのか、マルチモジュールで分けるのか等、考えなければいけないことは多くあります。	このセッションでは、いくつかの仮のアプリケーションをもとに、より深くアーキテクチャについて考えます。	- アーキテクチャに関心がある人 - モジュール構成やpackage構成に困っている人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Mori Atsushi
Navigation ComposeにおけるUIの状態管理	Navigationコンポーネントを用いた画面遷移の基盤構築は、Jetpack ComposeにおいてもNavigation Composeによってサポートされています。 特定のComposableをNavigationGraphのDestinationとすることで、そのComposableを一つの画面とみなした形での画面遷移を実現することができます。 Jetpack ComposeでのUI構築に限った話ではないのですが、UIの状態をどのようなスコープで管理するのかは、アーキテクチャ全体の構成に影響を与えるポイントです。 複数画面共通で保持して更新させたい状態や、画面をとじたら破棄したい状態など、要件に応じた使い分けが必要になることもあると思います。 そういった状態管理などのように実現させるのか、Navigation Composeを用いた場合の事例を紹介します。	- Jetpack Composeに興味がある方 - Navigation Composeをもちいた画面構築に興味がある方 - Androidアプリの新規開発を考えている方	25 minutes	日本語 / Japanese	Jetpack Compose	Yuto Akaike
This session submission has been withdrawn.						

targetSDK19の世界からtargetSDK33の世界に転生したけど何もわからない件	androidの開発環境は目まぐるしく進化しています。Coroutine、Jetpack、compose、KMMなどなどなどなど・・・・。 久々にandroid開発に復帰しようと思った際に環境が変わりすぎており何からキャッチアップすればいいのかわからない、という状況に陥ることが多いのではないでしょうか？ Jetpack全盛期からandroidエンジニアデビューした筆者が歴史と経緯を振り返りつつ、以前の実装では使ってたけど今はこれを使おう！を説明します！ お話しする予定の内容 ・非同期処理の変遷 AsyncTaskからRx、そしてCoroutineへ ・アーキテクチャの変遷 AAC ViewModelの登場 ・UIバインディングの変遷 ButterKnife.からDataBinding、そしてComposeへ ・画面遷移の変遷 Acitivityのみから自前Fragment管理、NavigationそしてComposeへ ・変わるものと変わらないもの	・昔はandroid開発をやっていて最近復帰しようかと思っている方 ・最近android開発を始めたがなぜそのような仕組みになっているのか気になる方	40 minutes	日本語 / Japanese	Android Framework	RyuNen344
エンタープライズ向け設定機能 managed configurationsとは何か	Androidアプリをエンタープライズ向けに提供する場合、アプリ開発者は企業のIT管理者がリモートでアプリの設定を変更できるような機能を搭載することができます。 これをmanaged configurationsといい、Google Workspaceのモバイルデバイス管理機能やEMM（Enterprise Mobility Management）と呼ばれるサードパーティ製の同様のサービスを通して利用することができます。 本セッションでは、managed configurationsがどのような仕組みで提供されているのかということについて、Android Enterprise関連のドキュメントやAndroid Frameworkのソースコードといった公開情報を元に詳しく解説します。 また、アプリ開発者がmanaged configurationsを利用した機能を実装する方法についても解説します。 アジェンダ（予定） ・Android Enterpriseの概要 ・managed configurationsとは ・managed configurationsをAndroidアプリに実装する方法 ・managed configurationsの仕組み	・Android Enterpriseに興味のある方 ・エンタープライズ向けアプリを開発したい方	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Yusaku Tanaka
2022年の動画再生アプリの作り方	2021年10月にJetpack Media3が発表されました。 これまでは別々のライブラリとして提供されていたメディア関連のAPIが単一のライブラリに統合され、動画再生に関する様々なユースケースを容易に実現できるようになりました。 また、近年ではピクチャー・イン・ピクチャーへの対応や折りたたみ式端末への対応など、動画再生アプリに対しての機能要件が多様化してきています。 本セッションでは、2022年に動画再生アプリを作る際に考慮すべきことや具体的な実装方法について紹介します。 予定している内容 ・Jetpack Media3とExoPlayerについて ・MediaSessionServiceを用いたバックグラウンド再生 ・ピクチャー・イン・ピクチャーへの対応 ・折りたたみ式端末への対応	・Androidのメディア系APIに興味がある方 ・動画再生アプリの品質向上に興味がある方	25 minutes	日本語 / Japanese	Jetpack	oidy
Android Management APIで学ぶ2022年のAndroid Enterprise	Androidデバイスは個人が利用する携帯端末としてだけでなく、企業が従業員に貸与する端末や店舗・オフィスなどに設置するKIOSK端末、特定の業務を遂行するための専用端末などビジネス分野でも広く普及しています。 こういったビジネス用途の端末を効率的に運用するためのソリューションとなるのがEMM（Enterprise Mobility Management）と呼ばれるサービスです。 EMMはAndroid Enterpriseとしてビジネスユーザー向けに提供されており、Google Workspaceのデバイス管理機能やサードパーティ製のEMMサービスなどで利用できます。 また、Androidは、EMMデベロッパーに対してGoogle Play EMM APIとAndroid Management APIという2つのGoogle Play servicesで利用できるAPIを提供しています。 そして現在、Google Play EMM APIはEMMデベロッパーの新規登録を終了しており、事実上Android Managemet APIを利用することが推奨されています。 本セッションでは、Google Play EMM APIからAndroid Management APIへの変化を通して、Androidがどのようにしてビジネス用途のデバイスを管理しているのか、企業のIT管理者はどのような管理機能を利用できるのかということについて解説します。 また、個人のAndroidアプリ開発者が利用可能な範囲でAndroid Management APIを体験する方法についても解説します。 アジェンダ（予定） ・Android Enterpriseの概要 ・Google Play EMM API と DevicePolicyManagerの関係 ・Google Play EMM APIからAndroid Management APIへの変化 ・Android Management APIを個人のAndroidアプリ開発者が体験する方法	・Android Enterpriseに興味のある方 ・DevicePolicyManagerやDevice Ownerなどデバイスをユーザーとして利用したことのあるAndroidアプリ開発者	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Yusaku Tanaka
デザインシステムと一緒にはじめるJetpack Compose	REALITYでは今年から本格的にJetpack Composeを採用し、既存画面のJetpack Composeへの置き換えや新規画面のJetpack Composeでの実装に取り組んでいます。 また、同じタイミングでデザイナーやiOS、Unity、Webを巻き込んだデザインシステムの構築も走り始め、デザインシステムを考慮しながらのJetpack Composeへの移行や、すでに実装したJetpack Composeへのデザインシステムの適用など双方向で意識した開発が必要となっています。 デザインシステムかJetpack Compose、必ずしもどちらかが整備された状態でスタートするではないと思います。そういった事例の方にも今回の経験が活かせればと思い、お伝えさせていただきます。 予定している内容 ・デザインシステムから考えるJetpack Composeの構成・移行 ・他チームとの協業を意識したデザインシステム ・デザインシステムとComposeの並行した立ち上げについて	・デザインシステムに興味がある方 ・Jetpack composeの開発に興味がある方 ・デザインシステム×Jetpack composeの立ち上げに興味がある方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	ぴか
バックグラウンド処理から学ぶAndroidにおけるFlutterのネイティブ連携	Flutterは、Dart言語で書かれたアプリコードをコンパイルし、Flutter Engineと呼ばれるランタイムで実行することで様々な処理やUI描画を行う、マルチプラットフォーム向けのアプリ開発フレームワークです。そのエコシステムは成熟しており、アプリ開発で必要となる機能・処理の大半は、3rdパーティーから提供されるライブラリを組み合わせて使うことでAndroidプラットフォームのことをほとんど意識することなく実装できます。 ただ、Flutterアプリで実装しようとすると苦労したり、意図した通りに動かなかったりする処理というものも存在しており、バックグラウンド処理がその一例として挙げられます。通常のAndroidアプリに実装するときにも様々な知識・配慮が求められるバックグラウンド処理ですが、Flutterを使う場合は、それに加えて、他のプラットフォームでの動作や、AndroidにおけるFlutter Engineのライフサイクルも意識する必要があります。 本セッションでは、バックグラウンド処理の実装を題材にして、Androidアプリにおけるflutterの動作や、Androidネイティブ部分から見たFlutter Engineの扱い方、FlutterとAndroidネイティブ部分の連携について説明します。 Flutterを利用されている方はもちろん、利用したことがない方でもAndroidアプリの開発経験があれば理解できるように説明します。Flutterでのアプリ開発で苦労する点や、それをAndroidエンジニアとしてどう取り組むことになるかの一例としていただければ幸いです。	・Flutterを使ってアプリ開発をされている方 ・Flutterに興味があり、UIの開発以外でどういう苦労があるのかを知りたい方	25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	kafumi
通知チャンネルに対応する	通知はアプリの外でユーザーとコミュニケーションができる重要な機能です。しかし、ユーザーに通知を受け取ってもらうハードルは年々高くなっています。 通知機能にはさまざまなアップデートが行われており、Android 8からOSで通知チャンネルを作成する機能が追加されました。細かく通知を管理する仕組みが提供されたことで、よりユーザーが通知の許諾のコントロールをアプリの内外でしやすくなりました。またAndroid 13では通知のPermissionが追加され、ユーザーが必要とする通知を選択できるようになりました。 本セッションでは実際に弊社のアプリで行ったチャンネル対応の事例を通して通知設定の課題・解決方法を共有します。具体的には次のような内容を想定しています。 - 通知チャンネルとは 　- なぜ通知チャンネルに対応するのか - 通知許諾とは 　- Android13と12以下の挙動 - チャンネル対応 　- 既存のアプリ内画面との両立 　- マイグレーションに当たっての課題 　- アプリ内の通知画面 　　- 表現方法 　　- ユーザー毎の設定 　- 実装上の課題 　　- 連打制御 - 効果検証 　- 許諾の変化 魅力的なサービスの情報をお届けするため、よりユーザーが使いやすい通知の設定を実現していきましょう。	通知チャンネル対応を検討している方 通知許諾を高めたい方	25 minutes	日本語 / Japanese	Android Framework	Miyabi Gouji
Jetpack Navigationの導入した際の困り事と解決策	業務でJetpack Navigationを導入した際の困りごとと解決策ついて紹介します。 ActivityとFragmentが混在するアプリをシングルActivity化したい時に役立つ内容になっています。 セッションで取り扱う内容 - 初期表示するFragmentを動的に変更する方法 - FragmentにThemeStyleを適用する方法 - Fragmentの遷移に合わせてステータスバーの色を変更する方法 - FragmentやViewよりも生存期間の長いViewModelを使用する方法	複数Activityのアプリをシングルactivity化したい方 JetpackNavigationを導入したい方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Sekioka

他人の「Androidアプリエンジニア」としてのキャリアに触れるために	「Androidアプリエンジニアって、どこに居るんですか？」と言われると、SNSや採用人事から聞いたことがあるフレーズではないでしょうか？この言葉が指し示すように、以前も、現在もAndroidアプリ開発を行っている各社における「Androidアプリエンジニア」の採用ニーズは非常に高い状況です。 そのような非常に採用への競争が加熱している状況で、各社・各プロダクトが要求するような人材ニーズを満たすためにはJob Description、採用広報のような手段を用いて、以下のような要素を始めとする情報を伝えることが重要と考えられます。 - 企業・プロダクトそのものの魅力 - それらが抱えている技術的な挑戦 - 企業が提供する事ができるキャリアプラン - ターゲットとなるクラスターへの適切なスカウト・ブランディング 一方で、実際に採用活動をするとなるとよく思いつくであろうカジュアル面談や面接を始めとして「他人のAndroidアプリエンジニア」としてのキャリアに触れるという側面があり、やはり簡単に取り組めるようなものではない、という気持ちはどこかにはあると思います。 そんな中で、採用成果をより向上させるため、現場で働くアプリエンジニアとしてどのような動きをすればいいのでしょうか？また、ひとえに「採用」といっても、中途・新卒の区分の違いや採用に至った後の育成・成長といった部分まで様々なステージがあるはずですが、それぞれでどんな行動が求められるのでしょうか？ 本セッションでは、6年目の中堅Androidアプリエンジニアが中途採用・新卒採用双方の目線で行動してきたアクションをベースに「Androidアプリエンジニアが行う採用活動」という、答えの出ない他人のAndroidアプリエンジニアのキャリアへの触れ方について考察・提案します。 このセッションを受け、会社での採用活動を見つめ直す機会を作っていただいたり、実際に行動する きっかけとしていただければと思います。 ※ このセッションは登壇者の個人の視点として語られるものです。所属する企業・団体に依存した内容は極力 演しますが、ある程度の偏りが残る可能性があることにご理解いただけますよう、よろしくお願いいたします。 <アジェンダ> - Androidアプリエンジニアの採用ニーズについて - 採用ニーズを満たすためにエンジニアとしてできることとは？ - Androidアプリエンジニアとしてより効果的なアクションとは何か？ - 中途採用と新卒採用における考え方や姿勢の変化について	- Androidアプリエンジニアの採用をやっている人 - 人事の採用活動に協力したい、と考えている人 - チームにAndroidアプリエンジニアを増やしたい人	40 minutes	日本語 / Japanese	その他 (Other)	Yoshihiro Wada
マイナンバーカードで電子署名する方法	業務でマイナンバーカードを使用してe-taxに対応した電子署名をするアプリを開発した際に、資料が見つからず大変だったので、得られた知見を共有します。 セッションでは、AndroidのNFCの基礎と、マイナンバーカードを使ってファイルに署名する方法を説明します。 1. マイナンバーカード関連の仕様書の入手方法 2. Android SDKの NFC APIについて 3. マイナンバーカードのデータリンク層の通信仕様 4. マイナンバーカードのアプリケーション層の通信仕様 5. 電子署名の入力データ作成方法（XMLの正規化処理） 6. マイナンバーカードにファイルを送信して電子署名する方法	e-tax、マイナンバーカードに興味のある方	25 minutes	日本語 / Japanese	ハードウェア (Hardware)	Sekioka
未経験Androidエンジニアが3人も増えた話	弊社では、今年になってAndroidチームが4名増えましたがそのうちの3名(自分含め)がAndroidがほぼ未経験でした。 それぞれのJoin体制やどんな研修・タスクを実施してきたかなど、振り返ってみての良かった点や改善点などをJoinした側からの目線で発表させていただきます。 受け入れ側・未経験者側の両方の方の参考になれば幸いです。 予定している内容 ・未経験Androidエンジニアの Join事例 ・未経験Androidエンジニアが取り組んで良かったこと ・未経験Androidエンジニアが取り組みべき課題・タスク ・未経験Androidエンジニアが嬉しい受け入れ体制	・未経験のAndroidエンジニアのJoinや採用を考えている方 ・未経験からAndroidへ転向を考えている方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	ぴか
FluxかMVVMか	現在のAndroid開発はMVVMアーキテクチャが推奨されており、多くのドキュメントも存在します。 しかし、場合によってはこれが最適とは言えないこともあると思います。 例えば、特定の画面に様々なデータを使っており、画面の状態が複雑になる場合、その設計も複雑になることが多いです。 このような課題を踏まえて、ピクシブ株式会社の弊チームでは一部Fluxアーキテクチャを採用しました。 その例を紹介しつつ、Fluxアーキテクチャ とMVVMの違いやそれらの優位な点を紹介していきます。 内容 - Fluxアーキテクチャとは - 弊チームでの実装例の紹介 - 実際に使用してみての使用感 - MVVMとの比較 - それぞれのアーキテクチャとどう向き合っていくと良いか	- Androidアプリ開発のアーキテクチャについて日々悩んでいる人 - アーキテクチャについて考えだした人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	gatosyocora
Re:Kotlin Nativeってandroid向けにも書けるんですよ	KMMでiOS向けに使われる、そんなイメージをKotlin Nativeにもっていないでしょうか？実はもともとandroid NDK用にもKotlin Nativeは使えます。別イベントでお話した内容の詳細版になります。 お話しする予定の内容 ・What is Kotlin Native ・Inside Kotlin Native ・New Memory Management of Kotlin Native ・interop other library	・C++あまりわかっていないがNDKにかじりつきたい人 ・Kotlin Nativeを用いてすべてKotlinで書きつつもすべてネイティブバイナリのアプリを作ってみたい人	25 minutes	日本語 / Japanese	Kotlin	RyuNen344
Android12L マルチスクリーン対応	以前Android3系(Honeycomb)でもタブレット向けの機能が提供されていましたが、時間が経ち忘れられていたかと思います。 そんな中Googleが2022年後半にタブレット、フォルダブル 端末向けに最適化されたOSAndroid12Lを提供することを発表しています。 Surface Duoや Chrome bookなども対応すると言われておりより一層タブレットへの需要が増える可能性があります。 また、Windows Subsystem for AndroidにてAndroid12Lが利用可能になるともアナウンスされており、これまでWindowsでは利用できないアプリ/サービスがAndroidが利用可能になったことにより、アプリを通して価値を提供できる気合いが増えるのではないかと考えられ、 どのような対応と、操作面、デザインにしていった方が良いのか発表させてもらえればと思います！ - Android12Lについて - Jetpack WindowManager/SlidingPaneLayout - 既存のアプリの対応すべきこと - 画面のライフサイクルについて - どんなデザインにすれば いいのか ~フォルダブル向け~ - どんなデザインにすれば いいのか ~タブレット向け~ - Windows Subsystem for Androidにてどのように動作させるのか 　- マウスやキーボードへの対応 　- AmazonのAmazon Appstoreについて(既存Google Play から移行させる方法)	Androidアプリ開発に従事している方々	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	@yass
Gradle再入門〜Gradleを制する者はandroidを制す〜	android開発で必須なGradleを再入門しようというセッションです。雰囲気でapplyして、kotlinで書けるらしいからgradle.ktsにしてみて・・・・・・などなどgradleはとても奥が深く、また非常に便利です gradleを理解すれば簡単なスクリプトをgradle taskとして実行させることで開発効率をグンと上げることができます 今一度再入門して開発効率を向上させましょう！ お話する予定の内容 ・gradleとは ・gradle taskとは ・gradle pluginまわり ・gradle.kts ・gradle pluginとgradle taskを自作する	・雰囲気でgradleを書くのを脱したい人 ・普段の開発のちょっとした細かい作業の自動化をgradleで行いたい人	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	RyuNen344
Unityを組み込んだAndroidアプリの作り方	モバイルアプリでARやVRなど3D表現をするアプリを作る場合、Unityが使われるケースが多いですが、Androidのネイティブアプリに Unityを一部の要素として組み込むことが可能です。 Unity as a Libraryによるライブラリ化や、アプリへの導入の仕方、チームでの開発における CI/CD環境について、実例を交えてご紹介します。	AndroidアプリにUnityを取り入れてみたい人 Unityをライブラリとして組み込んだアプリの CI/CD環境の実例を見たい人 自作のライブラリを CI/CD環境に組み込みたい人	25 minutes	日本語 / Japanese	その他 (Other)	tkc
Composeで地図を表示する	Compose導入を進めている中で、Maps SDK for Androidを使っている場合には 地図表示についてもComposeへ移行することを検討しているのではないでしょうか？ 相互運用APIを使って移行する方法もありますが、 直接、相互運用APIを使わずにMaps SDKをComposeを扱えるMaps Composeライブラリを用いて 既存地図表示をComposeへ移行する方法について解説します。	- 既存の地図表示をComposeへ変更していきたい人 - Composeを使っているアプリで地図を表示したい人	25 minutes	日本語 / Japanese	Jetpack Compose	Sofue
どうする？Jetpack ComposeのTheme ~ 基礎から既存アプリへの導入まで ~	Jetpack ComposeのThemeは独自のデザインシステムを構築できる等柔軟で強力な機能である一方、Android ViewのTheme/Styleとは異なる点もあり、どのようにThemeの設定を行えば良いか悩むのではないでしょうか？ 特にJetpack Composeを既存のアプリ開発に導入する際のThemeの設定は、既存アプリがMaterial Designのデザインシステムに沿っていない場合もあり、その中でどのようにThemeを設定し開発を進めていくかはJetpack Composeを導入する上で重要なポイントになります。 本セッションでは、Jetpack ComposeのThemeの基礎知識から拡張方法、既存のアプリへJetpack Composeを導入していく中でThemeをどのように設定し、普段の開発を進めていくかを実例を交えながら解説したいと思います 。 想定する内容 ・Jetpack ComposeのThemeの基礎知識 ・Jetpack ComposeのThemeの拡張、独自Themeの作成方法 ・既存アプリのTheme/StyleをJetpack ComposeのThemeとしてどう扱うか	・Jetpack ComposeのThemeについて知りたい方 ・既存アプリにJetpack Composeの導入を考えている方 ・既存アプリにJetpack Composeの導入をしたがThemeの扱いに迷っている方	25 minutes	日本語 / Japanese	Jetpack Compose	Horie1024

私の好きなModifier関数	新しいUI開発ツールキットであるJetpack Composeが正式リリースされ、Androidアプリの UI実装の主流が移りつつあります。 Jetpack ComposeでUI実装を進めるには、これまでのAndroid Viewで UIを実現していた各種Viewクラスのプロパティやコールバック、スタイルやシェイプ定義の代わりに、主にJetpack ComposeのModifierを使って実装することになります。 Android ViewではViewクラスのプロパティではなくスタイルやシェイプ定義が必要になり、少し手間のかかったUIついて、Jetpack ComposeのModifierではシンプルな実装になる場合があります。 またJetpack Composeではプロパティやコールバックが提供されておらず、実現が難しかったUIもあります。 大規模アプリでJetpack Composeを導入している経験と、新規アプリを全てJetpack Composeで実装した経験をもとに、本セッションではJetpack ComposeのUI実装で必ず触れることになるModifierについて話せれば と思います。	・これからJetpack Composeを使い始める方 ・Android Viewが好きな方 ・Jetpack ComposeでのUI実装方法を模索している方	25 minutes	日本語 / Japanese	Jetpack Compose	Kohei Yamamoto
Bubblesを使ったチャットアプリを作ろう	BubblesはAndroid 11で追加された機能です。 Bubblesを使えば、よりインタラクティブなチャット機能をユーザーに提供できます。 Bubblesを使うためにはショートカット、通知、通知の会話やユーザーから許可を取るためのチュートリアルの表示など様々な要素が登場します。 本セッションではそもそもBubblesとは何なのか、そしてBubblesを使ったチャットアプリを作るには何を行えばいいかについて解説します。 Bubblesはリリースされてからしばらく経っていますが残念ながら採用しているアプリをほとんど見かけません。本セッションを通じてBubbles対応アプリが増えることを期待しています。	とにかく新しい機能が好きな人 Bubblesが好きな人 他のアプリが使っていないAndroidの機能を使って自分のアプリを目立たせたい人 通常の通知では満足できない通知マニア	25 minutes	日本語 / Japanese	Android Framework	napplecomputer
7年ものの巨大アプリに下タブ導入するための戦略を振り返る	今年、ナビゲーションドロワーが導入されていた巨大アプリをボトムナビゲーションを使ったアプリにリニューアルしました。 Activity が大半を占めるアプリに下タブを導入した際に、トップレベルになる画面の中で Fragment に移行することができない画面がありました。 その際、我々は共存する道を選びました。 ボトムナビゲーションと Activity のままとなっている画面をどのように共存させたかについて話したいと思います。 さらに Activity から Fragment に移行する際に大変だった部分や解決策などについても話せれば と考えています。 また、今回のデザインリニューアルはエンジニアから話を進めました。 その中でビジネス側の人とどのように進めていったかなどについても触れたいと思います。 具体的には、以下の内容を話す予定です。 ・ボトムナビゲーションの導入背景 ・共存する道を選んだ理由 ・実装 ・Activity から Fragment への変更対応 ・SDKバージョンが31未満の場合の Multiple back stacks 対応 ・通知バッジ ・ビジネス側の方とのコミュニケーション	・Activity で作られた画面が多いアプリに下タブを導入したいと考えている方 ・エンジニア主導でリニューアルを進めたいと考えている方、また、進める必要がある方	25 minutes	日本語 / Japanese	Jetpack	Fumiya Tani
Jetpack Compose の Preview を効果的に使って開発する	Jetpack Compose の登場により Android アプリの UI 構築は格段に効率的になりました。その中でも Preview 機能は Jetapck Compose の主要機能の一つであり、XML/Layout Editor と比較してとても柔軟で強力にサポートされています。特に Android Studio Electric Eel からはついに Live Edit がサポートされたことにより今後の Android アプリ開発の現場で Preview 機能はますます不可欠なツールとなっていくでしょう。 弊社サービス「クックパッドマート」でも Jetpack Compose 導入初期から Preview 機能を非常に重要視しており、Preview をフル活用するためのアーキテクチャや仕組みを構築したことで開発体験の改善に大きく貢献しました。 そこで本セッションでは Preview 機能をアプリ開発で利用する上での知見やポイントと、開発フローを加速するためのさらに踏み込んだ工夫について実例を中心に紹介します。 主なトピック - Jetpack Compose Preview の仕組み - 効率よく Preview を使うためのアーキテクチャ設計 - Preview の開発フローへの組み込みと運用 - Preview の発展的な活用	Jetpack Compose を使って開発している方	25 minutes	日本語 / Japanese	Jetpack Compose	YutoKoguchi (10llip0p)
式年遷 Target SDK Version	時は2022年、Androidアプリエンジニアたちは毎年訪れる新OSの対応に追われていた……。 「待ってください、その開発リソースは新機能開発のためのものなんです！」 「ええい、Target SDK Versionを移すことができなければ今後アプリの更新はなしだ！」 「どうか、どうかそれだけはご勘弁を……！」 Google Playに対象APIレベルの要件が追加されたことで、Google Playで配信するアプリは毎年の新OS対応が必須となりました。 新OS対応は新OSのドキュメントを読んで対応するだけではありません。 - alpha版、beta版ででアプリにとって影響度の大きい変更を確認し、変更をリクエストする - 開発リソース、QAリソースの確保 など、リリースするためにはいくつものタスクがあります。 本セッションでは毎年襲われる新OSに具体的にどう対応していくのか、タクシーアプリGOのAndroid 12対応、Android 13対応をもとに実例を交えながら解説します。	Google Playでアプリを配信している人 新OS対応をやったことが無い人 他社の新OS対応が気になる人	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	napplecomputer
ニュースアプリにおけるRecyclerViewの設計と戦略	皆さんのアプリでは、RecyclerViewを使っていますか？ 昨今のUI事情的にはJetpack Composeの話題が持ち切りですが、Android Viewで作成するにあたってRecyclerViewはまだまだ最有力の選択肢の1つといえるでしょう。 RecyclerViewは公式APIや便利なメソッドが充実しており、表現力の高いUIを表示できることに加えて、View再利用時のパフォーマンスが非常に高いです。 しかしながらアプリ内の複数の画面でRecyclerViewを利用したい場合、Fragment-Adapter-ViewHolder間のスマートな設計について考えなければなりません。 特にニュースアプリのようなユーザーに情報が伝わりやすい画面を実現したいアプリではComponentが複雑化することが多く、冗長な記述を避けるために共通化しつつ、機能仕様を満たす工夫が必要となります。 GroupieやEpoxyといったサードパーティーライブラリを利用すると、複雑な画面を少ないコード量で柔軟に表現できます。一方で、ライブラリの制約に強く縛られてしまうデメリットが存在します。 本セッションでは、最新のJetpackのRecyclerViewのAPIを用いた一覧画面の実装方針について解説していきます。先日弊社のアプリでは、主要画面をListViewからRecyclerViewへ置き換えました。サードパーティライブラリを一切使わない素のRecyclerViewで様々なユースケースを満たすための設計を紹介します。 さらに、ListAdapterやConcatAdapterなどのRecyclerView.Adapterを拡張したクラスの実用的な使い方についても合わせて説明します。 具体的には次のようなトピックについて発表します。 - LegacyなListViewからRecyclerViewへの移行 - ListViewとRecyclerViewのAPIの差異のどう乗り越えたか - 主要画面をRecyclerView移行するにあたって乗り越えたニュースアプリ独自の仕様による制約 - 記事のinView計測 - 広告表示 - ListAdapterとDiffUtilによる効率的なアイテムの差分更新 - ListAdapterのAPIの紹介 - isItemTheSame/isContentTheSameの正しい使い分け - ViewHolderおよびアイテムの戦略 - 各画面でどのようにアイテム共通化を図ったか - KotlinのDelegationパターンを用いた解決手法 - ConcatAdapterを用いたFooterの実装 APIの挙動や内部の実装について触れることを通して、よりRecyclerViewの見方に近づいていきます。これを機にRecyclerViewの実装を見直してみませんか？	- RecyclerViewの便利な使い方を学びたい方 - ListViewからRecyclerViewに移行したいと思っている方 - RecyclerViewの共通化の方針に迷っている方	25 minutes	日本語 / Japanese	Jetpack	Yuya Shinohara
Chatworkモバイルチームの今までとこれから	7年前まで、Chatworkのモバイルアプリは Titaniumによるクロスプラットフォーム開発を行っていました。 それが6年前に、iOSとAndroidはそれぞれをネイティブ化し別アプリとしてリリースすることとなり、そこからプラットフォームごとのiOSチームとAndroidチームでそれぞれ開発がおこなわれてきました。 しかし、その間に以下2つの課題に直面しました。 1. iOS/Androidアプリのデザイン、機能、ユースケースの違い 2. リソースの偏り そこで、2021年2月にiOS/Androidアプリエンジニアが混在している2チームに分けるという、大きなチーム編成をおこないました。 混在チームから1年半たちましたが、お互いの機能を知ることでアプリの違いは徐々に減っています。 一方で組織やサービス拡大とともに新たな問題も出てきています。 それらの問題を解決すべく、今後目指していくチーム体制についてお話できればと思っております。	1サービスでiOS/Android両方のアプリをリリースされていて、アプリ開発体制やチーム作りに課題を感じられている方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	tinpay

Android "を" ビルドしてAndroid Systemを呪いてみよう	Android開発者の皆さんは日々Android アプリ (apk / aab) をビルドしていますが、Android"を"ビルドしたことがありますか？Andriodをベースとする組み込みデバイス等の開発に従事している方は、日常的にビルドしているかもしれません。 Androidはオープンソースでコードの大部分が公開されており、Androidそのものをビルドし、システムイメージを作ることが可能になっています。また、カスタムのカーネルをビルドし、それを前述のシステムイメージに埋め込むことも可能です。 システムイメージ・カーネルをビルドすること自体は普段のサービスアプリ開発において、役に立つ機会は少ないです。しかし、ただ何となくみていたAndroidシステムの挙動の裏側を鮮明に理解することができ、今後のアプリ開発に役に立つ機会がきっと来るでしょう。 本セッションでは、サービスアプリ開発者であるスピーカーの視点からまにアプリ開発者に向けて、Androidのシステムイメージのビルド方法、カーネルのビルド方法を Android Developers に記載されている手順を補完しながら step by stepで進めていきます。 その後、コードの一部を修正しながらAndroidシステムの動きに迫っていきます。 <予定しているアジェンダ> - Android システムについての簡単な概要 - Android のビルド方法 - カーネルのビルド方法 - 実例1. Android システムを修正しながら確認する navigation bar の振る舞い - 実例2 :Android システムを修正しながら確認する Dialog の振る舞い	Androidのシステムそのものに興味がある人 Androidをビルドしてみたい人 Androidのビルドに挫折した過去がある人	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Kazuki Chigita
アプリエンジニアとQAチームがデバッグ機能の改善に取り組むぞ！	アプリのデバッグ機能を作る際に、エンジニアがデバッグのために作る場合と QAやPdMチームなどの要請によって、作る場合があると思います。 サービスの運用が長期化されて、デバッグ機能が乱立し、 何がどう動いているデバッグ機能なのかのドキュメントの整理もままならない状態から、 QAチームが今必要としているデバッグ機能をリサーチ、 QA工数削減した、GOユーザーアプリ改善チームの取り組みを紹介します。 予定しているセッション内容 - GOユーザーアプリチームの開発体制 - GOユーザーアプリに求められたデバッグ機能 - 乱立したデバッグ機能を整理して、QAチームを今必要なデバッグ機能について考える	- QAチームがあるサービスを開発するエンジニア - デバッグ機能を実装している、実装したいエンジニア	25 minutes	日本語 / Japanese	開発体制 (Development Process)	Satoru Komai
Canvas Composable入門	これまでAndroidで既存のWidgetで用意されていないような独自の描画をしたい時はCanvas APIを利用してきたと思います。 Jetpack ComposeにはCanvas Composableが用意されており、これを使うことでカスタムグラフィックやアニメーションを簡単に描画できるようになります。 このセッションではCanvas Composeを使った描画のやり方を、実際にプロダクションにリリースした例を交えてお話しします。	Jetpack Composeで独自グラフィックを描画したい人 Jetpack Composeでアニメーションさせたい人	25 minutes	日本語 / Japanese	Jetpack Compose	tkc
移り ゆくデファクトスタンダードにチームとしてどう追従するか	モダンと呼ばれるような技術・設計は開発における生産性や諸リスクに有効であることは周知の事実です。 一方でそれら理想を導入するための障壁を感じてしまい、プロダクトコードが未だレガシー環境であることに悩まされている方も少なくないと思います。多くの場合は、モダンとレガシーとのギャップが大きいが故に導入に至るまでのロードマップ策定が行えない場合が多いと感じます。 MVVMをはじめとしてマルチモジュール化からJetpackCompose導入まで、導入を目指すには少なくない考慮が必要ですが適切なステップを踏むことによって、着実にそして確実にモダン環境への移行を進めていくことが可能です。 レガシーからモダンへと移り変わるための各種ガイドラインの策定とロードマップの作り方について、個人としてではなく「Androidチーム」として進められるよう、知見を実体験とともに解説したいと思います。	Androidチームとして技術的負債の返却、レガシーからの脱却を考えている方	25 minutes	日本語 / Japanese	開発体制 (Development Process)	tk-masuda
完全独学 -モダンなAndroid開発者への道のり-	私は、Web系ベンチャーで働き始めて1年のAndroidエンジニアです。 前職では全くの異業種・職種で働いておりましたが、1年の学習期間を経て、Androidエンジニアとしてのキャリアをスタートさせました。Androidは体系的なカリキュラムが世の中にあまり存在せず、独学での学習に苦労している方が多いのではないでしょうか。 このセッションでは、1年間スクールやメンターを利用せずたった1人で学習を進め、Kotlin製のMVVM + Coroutine + Firebase + Dagger Hilt を取り入れたAndroidアプリを実装した私の経験をもとに、Web系のベンチャーでも通用するモダンな開発スキルを獲得するための知見をみなさんに共有します。 詳細については以下通りです。 - 学習計画の立て方 　- 目標設定（避延評価学習法について） 　- 学習順序の決め方（プログラミング全般に必要な知識とAndroid固有の知識を効率よく身につけるためには） - 情報収集の仕方 - アプリ実装中に詰まった時の立ち上がり方 - Android独学のTips(おすすめ書籍、オンラインコンテンツなど) 加えて、1年間業務でAndroid開発に携わったからこそ見えた独学のメリット・デメリット、そして当時の自分に足りていなかった要素もお話しできればと思います。	- 未経験からAndroidエンジニアになろうと考えている方 - 市場にAndroidエンジニアが少ないことに課題意識を感じている方	25 minutes	日本語 / Japanese	その他 (Other)	tsemb012
デザインシステムを使って複数アプリのテーマを管理しよう	アプリのデザインを作る上で、見た目や操作感を統一してユーザーが使いやすくするために存在するのがデザインシステムです。Androidでは Material Design や Material You が広く使われています。 私の所属する会社では、現在社内標準デザインシステムを作成し、Web・アプリともにデザインを統一していプロジェクトを進めています。しかし、複数あるサービスごとにブランドカラーなどが分かれており、ひとつのデザインシステムだけでは対応できない場合が存在します。 本セッションでは、どのようにしてAndroidアプリにデザインシステムを導入していくか、アプリ別にどのようにしてテーマを用意して適用していくなど、「複数のアプリにデザインシステムを導入していく方法」についてをご紹介します。 【アジェンダ（予定）】 ・デザインシステムの紹介 ・Androidアプリへのデザインシステム導入について ・アプリごとのテーマ適用 ・ダークテーマへの対応	・デザインシステムに興味のある方 ・複数のアプリでのデザインについて興味のある方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	consomme
クルマとスマホとアーキテクチャ	最近Androidでは、スマートフォンやタブレットだけでなく多様なデバイス間でのシームレスなユーザー体験が注目されており、自動車もそのうちの一つです。 シームレスなユーザー体験の実現にはプラットフォームから提供される機能の活用だけでなく、アーキテクチャレベルでの検討・考慮が重要なのは言うまでもありません。 しかしながら、Android Autoをはじめとする車載機器と連携するプラットフォームを利用したシステムのアーキテクチャ、設計に関する議論や情報はあまり見掛けない印象です。 本セッションでは車載機器と連携するシステム開発に携わってきた中で考え、実践してきたアーキテクチャや設計、考慮しておきたいポイントなどを共有したいと思います。 内容(予定) - Presentation層で考慮したこと - どうしてUIとそれ以外を分けるのか - Repositoryパターン - クライアント部分 など	- スマートフォン単体以外でも動作するアプリの設計に興味ある方 - 自動車内外とでシームレスな機能を提供したいとお考えの方 - 車載機と連携するシステム開発に携わっている/携わる予定の方 - Service上でUIを構築するアプリ開発の仲間が欲しい方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	nyafunta9858
はじめようビルドメトリクス	オレはいったいどれほどの時間をこのプログレスバーと共に過ごしてきたんだろう.... Android開発とビルドは切っても切り離せない関係です。 このセッションでは手元のマシンから収集できる情報を集めて、どれほどの時間を費やしているのか可視化する方法を指南します。 - 1回の分析 　- 「ビルド」定義の確認 　- gradle-profilerを使ってみよう - 1日の分析 　- 簡単なAPIを作成してのビルド回数などの収集方法 このセッションでは「日常のビルドにかかる時間を知る」ことにフォーカスしているため、分析結果を使った高速化の方法などには触れません。	- 新しいPCを買ってもらおうとしたら上司に「本当にそんなスペック必要なの？」と言われてしまった人 - チームのスループットをあげるためになんでもしたい人 - ただ己の時間が何にどれほど費やされているのか知りたい人 - 「gradle-profilerのことは知らないけどなんか面白そうだな」ぐらいの人 Androidアプリをビルドしたことがあれば前提知識は不要ですが、gradle-profilerの深い話や高速化のノウハウなどはありません。	25 minutes	日本語 / Japanese	開発体制 (Development Process)	こまたつ
タクシーアプリ「GO」の機能開発と一緒に進めるDIライブラリ移行 (Koin to Hilt)	DIライブラリKoinを採用した大規模アプリでDagger Hiltへの移行を進めています。 アプリの規模が大きくなると、主要部分で使っているアーキテクチャやライブラリを変更するための検討が必要になります。 変更による影響範囲が広く、動作確認に多くの時間が必要になり、機能開発のスケジュールに影響が出てしまい、内部改善がとても難しくなります。 この問題を避けるために、機能開発にあわせて段階的なDagger Hilt移行のアプローチを検討しました。 具体的には、モジュール単位でDagger Hilt移行を進めることで影響範囲をコントロールできるようにします。 コントロールされた影響範囲のみリリース前に動作確認をおこないます。 機能開発のスケジュールにあわせて影響範囲をコントロールすることで、段階的に内部改善を進める事が可能になります。 本セッションでは、大規模アプリについて機能リリースを進めながらDIライブラリを移行して得られた知見について話せればと思います。 話すこと ・大規模アプリの主要部分を変更する際に考えること ・内部改善を段階的にリリースするためのアプローチ ・KoinとDagger Hiltの共存と段階的な移行方法	機能開発を進めながら大きな内部改善を考えている人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Kohei Yamamoto

Jetpack Composeでつくる簡単なアプリ2選！ ~ WebViewアプリ、音声プレイヤー ~	1. Jetpack ComposeでWebViewアプリを作ろう！ Jetpack Composeには、WebViewはありません。 そのため、WebViewを使ったアプリをComposeで作るには、ComposeのAndroidViewを使って作る必要があります。 また、Pull to refresh機能やDrawerなどを入れようとするとAndroidViewとComposeをうまく組み合わせて作る必要があり、それぞれが正しく動作するように工夫する必要があります。 これからAndroidアプリを作りたいけど、最初はWebViewアプリで、徐々にネイティブ化してくのに今後を見据えてComposeで作っておきたい… そんなあなたに向けて、ComposeでのWebViewアプリの作り方を解説します。 2. Jetpack Composeで音声プレイヤーをつくろう！ Androidで動画や音楽を扱うためのライブラリとしてExoPlayerがあります。ExoPlayerには動画再生用にはPlayerViewというものが用意されていますが、音声再生のみのプレイヤーを作ろうとすると自前でUIを用意する必要があるのでそう簡単にはいきません。 メディアの管理や再生自体はExoPlayerに任せつつ、再生、一時停止、数秒戻る、数秒進む、音声再生進捗を表すスライダーのような簡単なUIを例にしたJetpack Composeでの音声プレイヤーの作り方を解説します。 特に再生進捗を表すスライダーについては、ExoPlayerで常に再生状況を監視しつつUIも更新し続けなければならないため、Composeでの状態管理も工夫しなければならないところがいくつかあります。 メディアプレイヤーなどの常に状態が変わり続けるような機能をJetpack Composeで作ろうと思っている方に向けて、詳しく解説します。	- Jetpack ComposeでWebViewアプリを作りたいと思っている人 - Jetpack Composeで音声プレイヤーを作りたいと思っている人 - Jetpack Composeで何かアプリを作ろうと思っている人	25 minutes	日本語 / Japanese	Jetpack Compose	syarihu
CameraXことはじめ	CameraXがStableになってしばらく経ちました。 ただ、まだCameraXに触れてない、Camera2との違いがわからないという人向けにCameraXの概要やCamera2との違い、CameraXでできるようになったことなどをわかりやすくまとめてお話しします。	CameraXについての新しい情報がほしい人	25 minutes	日本語 / Japanese	Jetpack	Yo Kukita
CameraX × ML Kit でパスポートOCR機能を実装	海外旅行を予約する上でパスポートは必須の情報ですが、自身および同行者の名前や生年月日やパスポート番号などの情報を入力することは、ユーザーにとって少なからずの手間となります。 弊社が開発する海外旅行の予約アプリでは、その手間を軽減すべく、パスポートOCR機能を開発しています。 実装にはJetpack Composeにも組み込みやすいCameraXとML Kitの組み合わせを採用しています。 CameraXはカメラ機能を手軽に開発できるようにしたJetpackのライブラリで、ライフサイクルにバインドして動作します。 ML Kitは機械学習を利用した開発を手軽におこなえるようにしたSDKで、以前はクラウドで動作するAPIと共にFirebaseで提供されていましたが、オンデバイスで動作するライブラリはML Kitとして独立しています。 弊社アプリのパスポートOCR機能では、CameraXのプレビューと画像解析のユースケースと、ML KitのText Recognition（文字列検出）を利用しています。 それらライブラリを用いてパスポート情報を取得する実装の内容や、精度向上のためにおこなっている取り組み等を紹介します。	- CameraXに興味がある方 - ML Kitに興味がある方 - UXの改善に取り組んでいる方	25 minutes	日本語 / Japanese	Jetpack	Yuto Akaike
2022年のBytecode Weaving ~Google、Transform API やめるってよ~	いくつかのライブラリでは、Androidアプリケーションのビルド時にBytecodeを書き換えることにより魔法のような機能を提供しています。このBytecode書き換えをBytecode Weavingと言いますが、Bytecode Weavingを実現する際にこれまで使われてきたTransform APIが近い将来リリース予定のAndroid Gradle Plugin 8.0で削除されることが予告されました。 本セッションではBytecode Weavingで何が実現できるのか、またTransform APIの何が問題だったのかを解説します。その後、Transform APIに代わって導入されるInstrumentation APIがどのように問題を解決しているかを具体的な使い方とともに解説していきます。 Bytecode Weavingはライブラリ提供者にとってとても強力な手段です。何ができるかを知っていることでライブラリが実現できることが格段に増えるので、ぜひこのセッションで最新の使い方を学び便利なライブラリを公開していただけたらと思います。	ある程度build.gradleはいじったことがあり、魔法のような機能を提供しているライブラリが裏でどのようなことをおこなっているかに興味がある方、またそのようなライブラリを作ってみたいと思っている方を対象としています。	40 minutes	日本語 / Japanese	開発ツール & サービス (Productivity and Tools, Service)	zaki50
プロダクトで安全にDataStore移行する	JetpackライブラリのDataStoreは、アプリ内のデータストレージのライブラリです。SharedPreferencesと同様にkey-valueでデータを格納することが可能ですが、SharedPreferencesより優れた点があり、DataStoreへの移行が勧められています。 しかし実際に移行を考えると、なかなかモチベーションが湧いてこないかも知れません。 例えば、以下のようなことでモチベーションが低下しているかも知れません。 ・SharedPreferencesの使っているkeyが多すぎる ・移行しなくても不便がない また、マイグレーションガイドもあり、移行は容易そうですが、実際にプロダクトで移行を進めるためにはいくつか懸念があります。 ・SharedPreferencesでたくさんのkeyを用いているが、全てマイグレーションが可能なのか ・一部のkeyだけマイグレーションして、段階的に移行が可能なのか ・DataStore移行後にrevertした時にデータの不整合が起こらないか そこで、本セッションでは、DataStoreへの移行するモチベーション、移行方法、実際のプロダクトで移行した際に起こり得る懸念とその対応について紹介をします。 概要 ・DataStoreの概要 ・DataStoreに移行するモチベーション ・SharedPreferencesからのマイグレーションについて ・実際のプロダクトでの移行計画 　・段階的な移行 　・テスト・QC（QA） 　・DataStore移行後に起こりうる懸念 ・その他（可能であれば） 　・Preference DataStoreで保存したものをProto DataStoreでの保存に移行できるのか 　・SharedPreferenceのkeyの管理について	SharedPreferencesを使用したことがある方。 DataStore移行を検討している方。	25 minutes	日本語 / Japanese	Jetpack	Go Takahana
2022年ヘルスケアアプリのつくり方	健康管理を目的とした様々なヘルスケアアプリが開発されています。 ヘルスケアアプリでは加速のセンサーやカメラ、ウェアラブル機器などを使用しての拍数、運動時間、睡眠時間、摂取カロリーなど様々な種類のデータが扱われています。 このような健康状態や運動に関するデータは様々なプラットフォームで管理されており、 サービス自身が保持していないデータについてもプラットフォームを介して参照することで、 健康管理の体験を提供できるようになっています。 その一方で、Androidでは様々なヘルスケア管理プラットフォームが存在し、使っているサービスによっては異なるプラットフォームに計測データが保存される場合もあります。 また今年のGoogle I/Oでは新しいプラットフォームであるHealth Connectが発表され、複数の管理プラットフォームとのデータ連携ができるエコシステムとして期待されています。 本セッションでは、2022年に新しくヘルスケア アプリを開発した際に得られた知見をもとに、 ヘルスケア プラットフォームHealth Connectとの連携アプローチについて話せればと思います。 話すこと ・Google FitとHealth Connectの機能比較 ・iOSのHealth Kitとのデータ互換性を考える ・現時点で連携すべきプラットフォームとして検討した点	ヘルスケアに関わるアプリ開発に興味のある方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Kohei Yamamoto
複数事業、チームをまたいだ横軸組織「IUクロスプラットフォーム室」の挑戦と振り返りと今後について	組織体制として、一つの会社やグループ会社の中に、ネイティブアプリを開発する複数の組織が存在するというパターンはよくあります。所属している会社が大きくなるにつれて、もともとは1会社1プロダクト(=1ネイティブエンジニアの組織)だったところから、複数プロダクトや複数の子会社が生まれ、それらの組織の中に新たにネイティブエンジニアのチームが派生する、といったパターンもあるかと思います。 そのような複数組織が存在する組織の理想状態として、継続的な知見共有であったり、適切な人材流動といったものがあると思います。 本セッションでは、株式会社タップル、株式会社CAMなど複数の会社に所属するネイティブエンジニアが所属する仮想組織、「IUクロスプラットフォーム室」についてお話します。この組織は2022年4月に発足し、これまでIU管轄という管轄に所属する全ネイティブエンジニアが所属する仮想組織という位置づけで活動をしています。 このような複数の事業をまたいだ仮想組織を作成するに至った背景、理想状態、これまで行ってきたうえでうまくいったこと、失敗したことなどをお話し、最終的には今後の組織の理想状態を見据えた上で、どのようなと行っていく予定か、ということについてお話します。	複数の事業を跨いだ組織や、特定の職種に特化した横軸組織を作ろうと思っている方の参考事例になれば幸いです。	・複数の事業を跨いだ組織であったり、特定の職種に特化した横軸組織を作ろうと思っている方 ・複数のネイティブエンジニアの知見共有の仕組みなどを今後作っていきたい方	25 minutes	日本語 / Japanese	その他 (Other)	shoheikawano

READY for Compose!! -走りながら進めるJetpack Composeを受け入れるためのリファクタリング-	正式版リリースから早1年が経過し、急速に普及しつつあるJetpack Composeは、Android開発におけるUI実装のデファクトの地位を確立しつつあります。これは、Android開発の歴史の中でも有数のパラダイムシフトであり、すぐにでも導入したいと考える開発者が大半でしょう。
	
	しかし、世の中には多くの技術的負債を抱え、Jetpack Composeの導入をなかなか進められないアプリも数多く存在し、それらに携わる開発者の大きな悩みの種となっています。
	
	このセッションでは、リリースから6年が経過した弊社Androidアプリのリファクタリング事例を元に、Viewベースかつオレオレアーキテクチャで実装されたアプリをJetpack Composeが導入可能な状態にまで整備していく知見を共有します。
	
	Viewベースなアプリにおける Jetpack Compose導入の障壁をいくつか列挙し、それら障壁をどのように解決していったのか、具体例を提示しながら紹介できればと思います。特に、新規機能開発を並行で進めながら、アプリを段階的にJetpack Composeの導入が可能な状態へと近づけることに主眼を当て、息の長いアプリに携わる開発者の手助けとなるような内容を予定しています。
	
	### アジェンダ
	- Jetpack Compose導入の障壁
	  - Kotlinのバージョン
	  - Viewとロジックの密結合
	  - Viewベースの画面との連携
	- 障壁を乗り越えるために
	  - Kotlin syntheticsの撲滅
	  - Viewとロジックの分離(MVVM化や複雑なUIロジックの分離を段階的に進める Tips)
	  - Jetpack ComposeのViewベースのコードとの相互運用性をどう活かすか
		- Viewベースのアプリにetpack Composeを導入したい方
		- 新規開発のタスクに埋もれ、なかなかアプリのリファクタリングを進められていない方
			40 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Subroh Nishikori

アプリの大規模改修に合わせた Jetpack Compose 移行術	既存アプリへの Jetpack Compose の導入方法には、全て Jetpack Compose に置き換える方法と新規画面や一部の画面を Jetpack Compose に置き換えていく2パターンがあると考えています。 しかし、既存のアプリではプロダクト開発を進めつつ全画面を Jetpack Compose へ置き換えていく方法はあまり現実的ではありません。 そのため多くのプロダクトでは、新規画面や一部の画面から Jetpack Compose に置き換えていくパターンが多いかと思います。 私も後者の方法で Jetpack Compose に移行を行なっていますが、今回タイミング良くアプリの大規模改修がありそのタイミングで多数の画面を Jetpack Compose に移行しました。 また既存の navigation graph にも変更がありましたが、Compose 化した画面も含めて大規模に組み直しました。 このセッションでは、私がアプリの大規模改修に合わせて多数の画面を Compose 化しつつ画面遷移も組み直した経験を交えて AndroidView から Jetpack Compose への移行方法をお話しします。	- Jetpack Compose への移行作業に興味がある人 - AndroidView と ComposeView の両方を持つアプリの Navigation Component を用いた画面遷移処理に興味がある人	25 minutes	日本語 / Japanese	Jetpack Compose	nshiba
Kotlin Multiplatform Mobile と openAPI を活用して、アプリの品質と開発体験を向上させるまでの軌跡	皆さんは、アプリケーション開発中に、下記のような課題に悩んだ経験はありませんか？ - クラッシュやパースエラーなど意図しない挙動が多く、アプリの品質を向上させたい - アプリの品質を定点観測し、独自に SLI/SLO を定義し、リアルタイムに品質の変化に気付けるようにしたい - iOS / Android プラットフォーム向けにサービスを展開しており、API 通信ロジックを共通化させたい - クライアント側に複雑にビジネスロジックを保持しており、メンテナンスが辛い このセッションでは、モバイルアプリの品質と開発体験に焦点を当てて、既にサービスリリースされているアプリに、openAPI / BFF / Kotlin Multiplatform Mobile の仕組みを組み合わせて、これらの課題を解決するために取り組んだ際に得られたノウハウをご紹介します。 下記のアジェンダを想定しています。 導入に至った背景 - 弊社のサービス概要 - 従来の開発体制 解決したい課題と目的 - 課題の定義 - 目指す理想状態 - パースエラーによるユーザー影響の増大 - openAPI 導入によるコミュニケーションコストの削減 - openapi-generator を使ったクライアント、サーバー双方に向けたコードの自動生成 - Kotlin Multiplatform Mobile を活用して iOS / Android 間で通信ロジックを共通化しよう 手法 - BFF を段階的にリリースしていくために - openAPI を段階的にプロジェクトに導入していくために - 段階的リリースにおける、クライアント、サーバー側でのロールバック対応方針 運用 - BFF、openAPI を導入したことによって、どの様に開発体験が向上したか - 今後の課題とその解決手法 - チームへ開発体制を浸透させるために	- マイクロサービスとBFFの運用事例を知りたい方 - アプリケーションの品質の観測方法と、品質を向上させた事例を知りたい方 - 既存の大規模サービス (iOS / Android) にBFFとopenAPIの仕組みを導入していく事例を知りたい方 - Kotlin Multiplatform Mobile を活用した iOS / Android 間でのロジック共通化の事例とノウハウを知りたい方 - BFF、openAPI、KMM を組み合わせて開発体験を向上させることに興味を持っている方 - ネイティブ側にビジネスロジックを多く保持していてそれらのメンテナンスを簡単にしたい方 - 組織の開発スピードを加速させる事例を知りたい方	40 minutes	日本語 / Japanese	開発体制 (Development Process)	mzkii
Jetpack Composeの状態管理とAPIコール - React Hooksに習う、Apollo + Jetpack Compose	Jetpack Composeは、思想的にReactに近いところがあります。Composable関数とReactの関数コンポーネント、hooksはその最たる例と言えるでしょう。 また、GraphQLクライアントのApolloは、GraphQLの通信のレスポンスをグラフ構造そのままにキャッシュとその更新を可能にしています。Apolloのキャッシュ機構は、単なるGraphQL APIのコールのみならず、状態管理ライブラリとしての役割も果たします。これにより、フロントエンド開発者はAPIコールやキャッシュについて複雑に考えることが減りました。 しかし、Apollo KotlinにはApollo JSで用意されているのcustom hooksがないなどの差異により、React + Apolloのような開発者体験がJetpack Compose + Apolloの組み合わせでは得られません。 このセッションでは、React Hooksを参考にApollo Kotlinをラップし、シンプルにAPIコールとキャッシュ、状態管理を行うアーキテクチャを実現する方法について発表します。 - Jetpack ComposeとReactの類似性について - GraphQLとApolloについて - Apollo JSとApollo Kotlinの差分について - Apollo KotlinをJetpack Composeに適した形でラップする - ApolloとJetpack Composeで実現する状態管理アーキテクチャについて	GraphQLに興味のある人 Apolloに興味のある人 Jetpack Composeにおける状態管理に興味のある人	40 minutes	日本語 / Japanese	Jetpack Compose	mayamito
ComposeとKotlin Multiplatformを用いたクロスプラットフォーム開発	Composeは、もはやAndroidアプリ開発のためだけのものではありません。Compose Multiplatformの登場により、iOSアプリやデスクトップアプリ、Webアプリなどもcomposeを用いることでAndroidと共通化して実装することができるようになりました。 このセッションでは、Android, iOS, Desktop, Webをターゲットにした Kotlin Multiplatformのプロジェクトで実際にJetpack Composeを用いてクロスプラットフォーム開発する方法と、将来的にCompose Multiplatformを導入することを見据えたAndroid開発について発表します。 - Compose Multiplatformについて - なぜComposeがiOSやDesktop, Webで動くのか - Compose Multiplatformはどのくらい実用性があるのか - プロジェクトのセットアップ - コード共有について - Compose Multiplatformを見据えたAndroid開発の話	Androidアプリ開発でJetpack Composeを使ったことがある人 Kotlin Multiplatformに興味のある人 Kotlinを用いたクロスプラットフォーム開発に興味のある人 Compose Multiplatformに興味のある人	25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	mayamito
Flutterでもネイティブの処理がしたい！	最近ではクロスプラットフォーム開発がトレンドであり、実プロダクトで導入されている事例も多数見かけます。しかしながら、一部機能は共通化されておらず、ネイティブの言語で実装する場面もあります。 このセッションではFlutterでネイティブの処理が必要な際の実装方法、注意しておくべき点について解説していきます。	Flutterを少しでも開発したことのある方 クロスプラットフォーム開発に興味のある方	25 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	Motoharu Asanuma
アプリ起動時間短縮のためのBaseline Profileの導入と測定	□ 概要 アプリの起動時間はユーザーの関心を保ち、利用を拡大するための重要な要素です。 本セッションでは、実際に30%以上起動時間を短縮することができたBaseline Profileの概要や、既存のCloudProfileと比較した時のメリット、使用するユースケースなどを紹介します。また実際に行ったBaselineProfileの導入、測定、更新フローも紹介する予定です。 □ アジェンダ ・背景 　・既存技術(Cloud Profile)の紹介 　・Cloud Profileの問題点 　・Baseline Profileの紹介 　・なぜBaseline Profileを使うのか / 使わなくて良いケースは？ ・BaselineProfileの導入 ・BaselineProfileの効果測定 ・BaselineProfileの更新フロー	・アプリの起動時間のパフォーマンス向上に興味がある方 ・Baseline Profileに興味がある方 ・高頻度でアプリをリリースしている方	25 minutes	日本語 / Japanese	その他 (Other)	wata

モバイルアプリケーションの生産性を高めるCI/CDプラクティス	モバイルアプリケーションの生産性を高めるCI/CDプラクティス このセッションではアプリ開発を便利にするCI/CDの使い方やノウハウを学べます。 2022年現在、モバイルアプリ開発のワークフローは要件定義、リッチなデザインといった期待に応えるべく役割が増え、生産性の向上が求められています。組織の生産性が高い状態とはコーディングだけではなく、ソフトウェア品質やパフォーマンスなど非機能要件、デリバリーを通じて組織の要求水準を満たす必要があります。 CI/CD（継続的インテグレーション/継続的デリバリー）はこのようなプロセスを自動化を通じて整理できる良い方法です。ビルドやテストの自動化、QAにむけた社内デリバリーなど品質を保つ仕組みはもちろん、 Botによるlintチェック、ライブラリアップデートなどエンジニアにとって身近な問題も解決できます。 - 便利なビルド設定、Tips - Lintチェックやライブラリアップデートなどの自動化手法（生産性向上のためのプラクティス） - テストの自動化、環境に依存しないテストの書き方、アプリ設計 - QAや検証のためのデリバリー自動化 よく使われているクラウド型のGitHub ActionやCircleCI、Bitriseを対象としますが可能な限り、CI/CDプラットフォームに依存しない構成を目指します。 特定のCI/CDサービスを利用している・利用していない場合、どちらでも楽しめる内容に努めます。 CI/CDをメンテナンスする専門のロールがあるアプリ開発チームは少なく、開発が一息したときにエンジニアが改善することが多いのではないでしょうか。 このセッションでは聴講者がCI/CDを身近に感じ、改善のためのモチベーションを得られる状態をゴールと考えて、CI/CDのグッド・プラクティスを紹介します。	- CI/CDの導入を考えている方 - CI/CDを使いこなしたいがうまく使えてないと感じる方 - 自動化を通じて開発を快適にしたい方 - メンテナンスに課題を感じている方 - 組織的にソフトウェア開発を改善したい方	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	mhidaka
Android技術によるFlutter大解剖	FlutterはGoogleによって開発されているマルチプラットフォームフレームワークです。 いま注目を集めており、多くの企業で採用されています。 一方、XamarinやReactNativeなどマルチプラットフォームフレームワークは既にいくつか存在しますが、なぜいまFlutterが注目されているのでしょうか？ この問いに答えを出すべくFlutterの内部実装を紐解きます。 Flutterが他のマルチプラットフォームと何が違うのかを、Androidネイティブの技術をベースに解説致します。	Flutterやマルチプラットフォームに興味がお有りの方、Androidアプリの開発経験者の方に向けた内容です。 Flutterの知識、その他マルチプラットフォームの前提知識は不要です。 セッションの中で必要なものを解説致します。 セッションの中で登場するAndroidネイティブのコアな技術についても解説致しますが、一般的なアプリ開発の知識がある方を前提としたします。	40 minutes	日本語 / Japanese	クロスプラットフォーム (Cross-platform Development)	watanave
Deep dive into Jetpack Compose Text	我々Android Textチームは新しいUI Toolkitである Jetpack Composeのテキスト描画APIを設計し、実装しています。テキストは多くの部分をプラットフォームのAPIに頼っており、どこまでがライブラリ内の実装で、どこからがプラットフォームAPIを利用しているのかがわかりにくくなっています。この発表では最新のテキストに関連した Jetpack Composeの機能を、APIとその内部構造を中心に解説していきます。 Our team has been developing a new UI toolkit called Jetpack Compose. The Android Text team has been designing and implementing the Text rendering APIs in Jetpack Compose. Text components are still relying on lots of platform APIs, so it is quite hard to find a line between library implementation and platform APIs. In this session, I'm going to introduce the new feature in Jetpack Compose and what happens under the hood.	Jetpack Composeの内部構造に興味がある方 Androidがどうやって文字列を描画しているのか興味のある方 People interested in the internal details of Jetpack Compose. People interested in how Android draws text on the screen.	25 minutes	日本語 / Japanese	Jetpack Compose	Seigo Nonaka
What's new in Android Text 12 and 13	我々Android Textチームは次のAndroid OSにいくつかの新機能を実装しましたので、その紹介と使い方、内部の解説をしていきたいと思います。また昨年は、あまり解説できていなかったAndroid 12で追加されたテキスト関連の新機能と解説を合わせて、おこなっていきたいと思っています。Android 12では、主にTextShaper APIを用いた高品質なテキストアニメーションの実装方法を、Android 13では行間の調整方法と日本語に関連した新機能の紹介をさせていただいたいと思います。 Android Text released some new features into the next Android platform. I will explain the overview, how to use it and what is happening under the hood. Also, I will cover the Android 12 features in this session because I couldn't explain it last year. From Android 12, I will introduce the high quality text animation using TextShaper API. From Android 13, I will explain how to adjust the line height of the TextView, and explain the Japanese related new features.	最新のAndroid OSの機能に興味がある方 高品質なテキストアニメーションの作り方に興味のある方 People interested in latest Android OS features. People interested in how to make a high quality text animations	40 minutes	日本語 / Japanese	Android Framework	Seigo Nonaka
Compose 時代の開発ツールを整理する	Jetpack Compose の安定版がリリースされてから一年あまりが経ちプロダクション環境での導入事例もよく聞くようになってきています。 それに伴い Jetpack Compose の開発サイクルを手助けするための開発ツール、テストツールも新しいものが登場してきています。 このセッションではそれらのツールを整理し、開発サイクルのどの段階で使用するのが適切か考えていきます。 セッションで話す内容: - UI tests in Compose - Live Edit - Multi preview - Recomposition counts - Animation preview など	Jetpack Compose を導入した人 Jetpack Compose の導入を考えている人	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	thagikura
YouTubeライブ配信機能、実現への道 in 2022	昨今、オンラインのコミュニケーションはリアルタイム性をもとめられるシーンが多くなってきております。 動画配信プラットフォームなど顕著な例ですが、ライブ配信などを行う機能が提供され今では万人がそれを利用しています。 その流れはもちろんモバイルアプリにも来ており、ライブ配信機能を提供するアプリも増えております。 そこで今回はYouTubeを対象にAndroidアプリからのライブ配信を実現する方法をお話させて頂きます。 具体的には、 ・YouTube Data APIを利用したYouTubeへのライブ配信機能の実装方法 ・リリースなどするにあたって必要となる審査などについての対応 など2022年の現時点でAndroidアプリでライブ配信機能を実現するにあたって対応が必要な情報・フローをまるっと全てお話させていただきます。 キーワード - YouTube Data API - Google OAuth2.0 - Google verification - RTMP - Multi channel	・YouTubeへのライブ配信の方法・仕組みに興味ある方	40 minutes	日本語 / Japanese	その他 (Other)	yurihondo

Considerate App Update Delivery	リリースフローを自動化したことで安全なリリースだと思いこんではいませんか?
	
	自動化によって属人化やヒューマンエラーが減った冪等性のあるフローなら、確かにそれが動いている間は安全と言えるでしょう。
	しかし障害等で手動でのリリースが必要になったとき、その自動化されたフローから復元した手動リリースは安全ですか?
	
	安全なリリースフローとはヒューマンミスの排除に限りません。例えば required feature や dangrous permission の追加はアップデート配布対象が減るだけでなく、自動アップデートをオフにしてしまいます。それが意図的であったとしても、リリース前に妥当性及び副作用を確認出来ることが望ましいでしょう。
	
	多くのリリースフローは最小に近い手順を自動化しているだけで、上記のようなケース・検証の穴が十分にカバーされていないように思います。それらの検証が欠けているとき、そのリリースフローは安全とは言い切れないかもしれません。あるいはリリースよりもずっと前から継続的に行う検証が必要かもしれません。
	
	本発表ではAPK/AAB から静的に取得出来る情報、端末で取得出来る情報、そしてファイルからはどうしても取得出来ない情報を理解し、それぞれの情報を安全なリリースの実現やリリースフロー自体の改善に利用する試みを紹介します。
	
	- APK/AAB から静的に取れる情報
	- ランタイムで取れる情報
	  - FireOS などの例外
	- アプリファイルには含まれない情報
	- 情報の利用方法
	  - file metrics
	  - ライブラリアップデート時の proguard/R8 rules のレビュー
	
	※ いわゆるアプリの品質、QAについては言及しません。
		安全なリリースに興味があるひと
		proguard/R8 rule の変更について、実際に動かすことだけが唯一の検証だと思っているひと
		正直手動でリリースするときの方法を忘れてしまったひと
			25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	Jumpei Matsuda

GitHub Actions で構築する Android アプリの CI/CD	GitHub Actions は GitHub 謹製の CI/CD ワークフロー実行環境です。GitHub 上での開発フローと相性が良く、GitHub のマーケットプレイスでは様々な Action（再利用可能な部品）が公開されています。 私はこれまで開発組織において、組織を横断し GitHub Actions を利用して複数アプリの静的解析・テスト・ビルド・バイナリの配布といった CI/CD の環境を構築し、長らく運用してきました。本セッションでは、その構築・運用の内容や得られた知見、工夫した点、注意事項など説明します。 このセッションに含まれる話題 ・GitHub Actions でできることや利用する上での注意事項 ・静的解析 (Android Lint, ktlint) とボットによるレビューコメント ・テスト (Local unit test, Instrumented test) ・アプリのビルドと配布 ・Slack との連携、効果的な通知 ・独自 Action の作成 ・開発フローの自動化（コードによる GitHub の操作） ・各種トークンの管理 ・処理の共通化による再利用	皆様の CI/CD 環境構築の手助けになれば幸いです。	・これからCI/CDの導入を考えている方 ・自動化に興味がある方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	hkusu
作って遊ぶ Compose Animation	Jetpack Compose では、アニメーションを実装するためのさまざまなAPIが提供されています。これらを使うことで、簡単にリッチなアニメーションが作れるので、非常に感動モノです。 しかし、多様なAPI が存在するため、どれを使えばいいか迷った経験はありませんか？ちょっと込み入ったことをしようとしたときに途端に難しく感じてしまったことはないでしょうか。 本セッションでは、Jetpack Compose におけるアニメーションについて整理をし、どのようなことが実現できるかを考えます。実装の際に注意すべき点をみていきながら、Compose でのアニメーションに対する理解を深めていきます。	- Jetpack Compose を使ったことがある人 - Jetpack Compose でのアニメーションに興味がある人	25 minutes	日本語 / Japanese	Jetpack Compose	warahiko

Android のバッテリー管理	端末の進化に伴い、バッテリー容量も日々増えていっていますが、Android OS 自体にもバッテリー維持を目的としバッテリー運用を行う「アプリスタンバイバケット」という機能があります。 本機能は Android9 から追加された機能ですが、Android13 では更に新しい制限が追加されました。 本セッションではそもそもアプリスタンバイバケットとは何なのか、「制限付き」ステータスになった際にアプリにかかる制限、開発者が留意すべき点について解説します。	バッテリー使用量の多いアプリを開発している方 Android13 対応を行っている方 BOOT_COMPLETED の BroadcastReceiver を利用している方	25 minutes	日本語 / Japanese	Android Framework	sobachanko
人の声を可視化する	ひとの声を可視化するとき、波形型で表示するのが一般的です。 Android端末のマイク性能にばらつきがあることによって録音時の波形が端末によっては小さく、音声が入っている箇所が分かりづらかったり、一定数秒間に表示する波の数がずれることがあります。 人の声を可視化するオーディオビジュアライズについて、Androidで起こりやすい課題と対策を紹介し、またAudio処理に関わるアーキテクチャ、AudioRecordの取り回し方の注意点などについても ご紹介するつもりです。	こんな人におすすめ ・これからAudio処理を学ぶAndroidエンジニア ・人の声を取り扱うプロダクトに参画するAndroidエンジニア 受講者が得られる知見 ・Audio機能の基礎おさらい ・Audio処理を実装する際の注意すべき点と最適化手法 ・Audio処理まわりで端末依存における条件分岐を実装したことがある方	40 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	Miyuki Onuma
Modifierと仲良くなろう	Jetpack Composeで開発を進めるうえで必ず使うModifier。何者なのかをちゃんと理解して使えていますか？理解していなくてもなんとなく使えるけど、理解しておけばJetpack Composeがもっと好きになれるかも。 Modifierで実際に何が行われているのか、コードを読みながら仕組みを1つ1つ理解していくように解説していきます。そして、独自のModifierを自作することもできるようになります。みんなでModifierを完全に理解しに行きましょう。	Jetpack ComposeのModifierを使っているけど正直よくわかってないので理解を深めたいという方に聞いて頂けると、より良くJetpack Composeを利用できるになると思います。また、これから使おうと考えている方にもJetpack Composeの面白さを知っていただける機会にもなるので聞いて頂きたいです。	25 minutes	日本語 / Japanese	Jetpack Compose	makun
Android と Java の Reflection	Androidの実行環境がJavaなことは広く知られています。 しかし、Javaといっても、Javaにはバージョンがあり、その違いもあります。 本セッションでは、このことについてReflectionの観点から話したいと思いますが、まずはReflectionの紹介からしようと思います。 Reflectionって何？ JavaのReflection KotlinのReflection いろいろできるぞ、Reflection ・メンバー一覧 ・メソッド一覧 ・引数 ・引数名 ・アノテーション ・親クラス、etc.. さらにこんなことまで、できちゃうぞ！ ・インスタンス化 ・private なメンバーへのアクセス ・private なメンバーの呼び出し なんでもできるような気がしてきたReflectionですが、ついに制限が課されるようになりました。 ・Javaのバージョンによる Reflectionの制限 ・内部APIのprivateメンバへのアクセス制限 さて、Androidは Javaだと思っていますか？ 一口にJavaといってもJavaにもバージョンがあります。 Androidはどのバージョンの Javaなのでしょうか？ つまり・・・ 本編 ピュアJavaだと思った？残念Androidでした～いつからAndroidをJavaだと錯覚していた？ :Reflection編 AndroidのOSバージョンによるReflectionの制限 とはいえ、それ以外はJavaだもんね？・・・ Reflectionを使ってJavaでは失敗するのにAndroidでは成功する例 注意：このセッションは好奇心を満たすだけで、明日使えるテクニックは何もありません	AndroidやJavaの隅っこの方まで興味のある方。 好奇心を満たしたい方。	25 minutes	日本語 / Japanese	その他 (Other)	soranakk
Android Billing Library再入門	一言に「アプリ内課金」と言っても、課金から派生して考えるべきことは多いです。 例えば料金の変更や月額課金から年額課金への乗り換えなどです。 Android Billing Libraryでは様々な購入方法をサポートする他、これらのような購入に伴って発生する多くの作業を補助してくれる機能がありま す。 私はアプリ内課金の実装をする際、公式ドキュメント含め具体的な情報が少なく苦労しました。このセッションを聞けば「アプリ内課金を導入するにはこうすればよい！」の概要を理解できるようにお話します。 特に私が過去ハマった「定期購入アイテムの料金変更」に関しても注意点含めお伝えしていきます。 本セッションでは「そもそもアプリ内課金をどうやってアプリに導入するのか」から始まり、どのようなことがAndroid Billing Libraryでできるのか、発表者が過去ハマった難しい点などのノウハウをお伝えします。 以下のお話をする予定です。 - Android Billing Libraryでできること、その概要 - アプリ内課金（Android Billing Library）をアプリに導入する方法 - 選択できる様々な支払い方法 ・v3から可能になった現金支払い ・プリペイドプラン - お得な購入方法の提供 ・プロモーションコードや無料試用、お試し価格 - 定期購読を価格変更する方法と注意 点	- アプリ内課金を導入してみたい方 - アプリ内課金は導入済みで、次のステップに進みたい方 - お金儲けしたい方	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	furusin
Rethink Classic Bluetooth	In recent years, Bluetooth Low Energy (BLE) with its low cost and low power consumption has become a hot topic, but for the purpose of connecting Android devices and exchanging information bidirectionally, conventional Bluetooth ("Classic Bluetooth" in retronym) with its outstanding stability and wide bandwidth can still be useful. The development methodology of Classic Bluetooth is briefly explained on the official Android Developer website, but it is insufficient for creating practical tools and games. In fact, the information on the official website alone can be very confusing as to how to accomplish the following scenarios: - How do you send real-time game commands to each other while establishing a connection? - How do you send and receive large images and videos? - How do you retry if a send/receive fails or is incomplete? In this session, the presenter will go one step further than the official site to present practical Classic Bluetooth communication methods learned from past development experience.	This session is open to anyone interested in communicating using Classic Bluetooth. The content is intended to be a step further than the official site, but will still be something that people who are already using Bluetooth to create practical tools and games already know. Therefore, this session is more for beginners.	40 minutes	English	Androidプラットフォーム (Android Platform)	@fushiroyama
ついにアプリ毎の言語設定がOSレベルでサポートされたぞ！	アプリの多言語化対応を実施したことはありますでしょうか？ OS の言語設定に従って切り替えるだけで良いのであれば、各言語毎のリソースを用意し、適切にフォルダ分けをするだけで対応が完了します。 しかしながら「アプリ内で言語切り替えできるようにしてよ」と要求されることがままあります。 これまでアプリ内言語切り替え機能を提供するには OS バージョン毎に異なる複数のワークアラウンドを用意する必要があり、大変な苦労を強いられてきました。 しかし Android13 および AppCompat 1.6.0 においてようやく OS レベルでアプリ毎の言語設定がサポートされることになりました。 本セッションでは、過去の OS における独自の言語切り替え機能を提供する手法を解説し、その上で今回提供されたアプリ毎の言語設定が素晴らしいかを紹介しつつ、多言語化を実施する際の注意点を解説します。	アプリの多言語化を考えている方 独自の言語切り替え機能で消耗している方	25 minutes	日本語 / Japanese	Android Framework	sobachanko
アーキテクチャ検討の進め方	「開発においてアーキテクチャは大切」とは理解している方が多いと思います。 しかし「自分たちに合ったアーキテクチャ」や「新しいアーキテクチャへの置き換え方法・流れ」はどのように考えるのがよいでしょうか。 そもそも「自分たちに合ったアーキテクチャ」ってなんだろうか。考えた結果どうやって置き換えたらいいんだろうか、そう感じる方は多いのではないでしょうか。 SansanでEightを開発しているAndroidチームでは「アーキテクチャ検討会」と題して新たなアーキテクチャの検討・設計・導入方針の策定を進めています。 本セッションでは、その「アーキテクチャ検討会」の中で具体的に実施したことや難しかったこと、工夫した事をお話します。 その結果「自分たちでもできるかも！」と思っていただけるような内容にしたいと考えています。 以下のお話をする予定です。 - 「なぜ検討して置き換えなければならないのか」を腹落ちさせよう - 課題とやりたいことの交わるところが「やらなければならないこと」 - 検討だけでなく実践的に - どうやって置き換えていく？	- 長年メンテナンスしているアプリのアーキテクチャにモヤモヤしている方 - 新たなアーキテクチャに置き換えたい方 - 自分たちに合ったアーキテクチャを模索したい・している方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	furusin
タクシーアプリ「GO」から見る組織論 - 2つの異なる文化の組織が統合し、成長する組織となった方法とは？	株式会社Mobility Technologiesは、2020年4月に、JapanTaxi株式会社とDeNAの一部事業が統合し誕生した会社になります。 2020年9月に異なる2つタクシーアプリが統合し、1つのタクシーアプリ「GO」をリリースしました。 しかしながら、それぞれ異なる文化を持つ組織が統合し、1つのアプリを開発することは容易なものではありません。 また、同じ「GO」のユーザーアプリのプロダクト開発に従事するメンバーはPdMやデザイナーも含め40人を超えていたため、どうしても意思疎通がうまくいかなかったり、意思決定が遅かったりと、様々な問題を抱えていました。 そこで、2021年5月から、PdMやデザイナーも巻き込んだ、プロダクト開発チームとしてのマクロな改革を行いました。 当時考えていたこと、また、そこから見えた、合理的なチームの形を組織論としてお話できればと思います。 ・統合時の課題 ・開発体制の変更 ・チームの目線合わせとチームビルディング ・複数のチームをまとめる開発手法のフレームワーク ・そこから見えた組織論	組織の課題に悩んでいる人、これからチームを良くしていきたいリーダー	40 minutes	日本語 / Japanese	開発体制 (Development Process)	takahia
私の考えた最強のカーデザインUI/UX	音声アプリを例に、カーデザイン対応・改善した話を紹介します。 また、自動運転が近未来の街中で普通に使われるようになることを仮定し自動運転モードの場合のUIUXを紹介できればと考えています。 ディスプレイ・操作機器の細部にわたるデザイン・設計・実装	・カーデザインに携わったことがある方 ・既存あぶりをこれからカーデザインに対応しようとしている方 ・カーデザインのスマートフォンアプリのUIUXを設計したことがある方	25 minutes	日本語 / Japanese	UI・UX・デザイン (UI・UX・Design)	Miyuki Onuma

プロダクトへのJetpack Compose導入の課題と利点	Navigation Componentを利用し開発してきた実際にリリースし運用しているアプリケーションにJetpack Composeの導入をしました。その際、どのように導入を進めるのかやパフォーマンスをどのように担保するか、そのほかいくつかの技術的な問題などが多くありました。しかし、それらを解決させたことで今ではほとんどの画面をJetpack Composeを利用し開発できています。このセッションでは、具体的にどのような課題をどう解決してきたかを紹介し、Jetpack Composeの導入は十分実用性があり採用する利点が大きいことを伝えます。	これからJetpack Composeを導入しようと考えている方や、導入したいけどメリット・デメリットを考慮したうえで慎重に判断したい方、実際のプロダクトへの導入の雰囲気を知りたい方などに聞いて頂きたいです。そのほかにも、Jetpack Composeを個に興味のある方などにも聞いて頂けると、より興味がそそられるかもしれません。	25 minutes	日本語 / Japanese	Jetpack Compose	makun
Now in Android から学ぶ現代の Android アプリ開発のプラクティス	Android アプリ開発はトレンドが移り変わるのが早く、Framework, ライブラリ, アーキテクチャなどの開発手法もアップデートが早いためすべての変更についていくことは難しく、また特定のライブラリ構成、開発手法が全ての開発現場にとって最適であるということは難しくなっている状況になっています。 そんな中 Google から Now in Android という Jetpack Compose を用いた参照実装がリリースされました。 このセッションでは一つの推奨例として Now in Android の構成、使用技術を紐解いてそれぞれの要素がなぜ使われるようになったかを見ていきます。 話すことの予定: - Now in Android のモジュール分け - アーキテクチャ構成 - Hilt を用いたテスト手法 - Baseline profiler など	Now in Android の構成に興味がある人	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	thagikura

GooglePlay 定期購入実践 -無料試用と複数プランをリリースして-	このセッションでは、GooglePlay BillingLibraryを用いた定期購入を実装、運用してみて気づいたことなどについてお話します。
	
	ネイティブアプリでアプリ内課金機能を実装する際、Androidでは GooglePlay BillingLibraryを利用します。
	BillingLibraryは 単純に課金機能を実装できるだけでなく、例えば定期購入における無料試用期間などの、ユーザー取得に便利な機能も多数用意されています。
	また、定期購入のプランをユーザーが変更したい場合に、現在のプランで既に支払ってある金額と、新しいプランの金額とを うまい具合に 調整して支払いを行える比例配分モードも用意されています。
	メジャーアップデートも一年に1回行われていますが、今年の BillingLibrary v5では課金アイテムの在り方から刷新されるなど、とても大きい更新が行われました。
	利用している身としては、動向が非常に気になるライブラリの一つと言えます。
	
	そんなBillingLibraryですが便利な一方で、GooglePlayを介する性質上、特に定期購入はどうしても実装や運用に一手間が必要になります。
	
	例えばアナリティクスを取りたいという要望に対して、欲しいタイミングで必要な値が取れないこともあるかもしれません。
	比例配分モードも便利な機能ですが、モードによって返却値に差異があることもあります。
	実装時はOKだったとしても、BillingLibraryメジャーアップデートによってGooglePlayからの通知内容が変わることもあります。
	広かなことに、アプリで利用しているBillingLibraryのバージョンに依存しないこともあるでしょう。
	
	このセッションではアプリ内課金機能による定期購入の、そんな「実際やってみたら想定外」な事例を、アプリ内のコードも交えてお話したいと考えています。
	
	内容としては以下のように予定しています。
	- アプリ内課金定期購入について
	  - 基本的な処理の流れ
	  - 比例配分モードの簡単な説明
	  - v4 → v5 の変更点の簡単な説明
	- プロダクトの事例紹介
	  - 複数のプランと採用した比例配分モード
	  - 想定外だった事例
	  - 比例配分モードによる違い
	  - 無料試用と比例配分モードの干渉の一例
	  - 取れなかったアナリティクス
	  - v4 → v5のタイミングで変わった通知
		- アプリ内課金に興味がある方
		- GooglePlay で定期購入を提供しているサービスに関わる方
		- これからGooglePlay で定期購入を提供するかもしれないサービスに関わる方
		- 複数プランの定期購入を提供しようとしている方
		- GooglePlay BillingLibraryを用いて開発されている方
		- GooglePlay BillingLibraryのことを少しでも知りたい方
			25 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	nacatl

Androidチーム拡大の軌跡と、そこから学んだこと	ZOZOTOWNのAndroidアプリ開発に私が参画したのはおよそ7年前になります。 一時はAndroidエンジニアが私1人になるなどの危機に直面しながらも、現在は2チーム体制になるほどチームが拡大しました。 そんなZOZOTOWN Androidアプリチームの組織拡大までに遭遇した課題とその解決への取り組み、そしてそこで学んだことについてお話したいと思います。 これからチームを拡大しようと考えているみなさんの参考になれば幸いです。 以下の課題について当時の時系列にそって話す予定です。 課題1: シニアエンジニアの離職が相次ぎ、案件を回す力が衰える 採用活動を行うもシニアエンジニアの採用はできなかった。 課題2: 社内の別PJ解散に伴い人数が倍以上に増員。マンパワーの懸念は解消されるも、異なる文化のメンバーを一度に受けいれることに 週次の振り返りに加え、メンバーの提案でスクラムのエッセンスを導入。見積もりと同時に既存仕様や既存コードの解説を行う。 課題3: チームが安定し1人1人がパフォーマンスを出せるようになるも リーダーがボトルネックに メンバーが仕様検討に参加できるような制度としてオーナーシップ制度を開始。 課題4: 案件が多く回るようになると同時に、やりたいことが増加 採用活動強化のためカジュアル面談の改善、オンボーディングのマニュアル化、メンバーの情報発信強化に取り組む。 課題5: 採用活動がうまくいくも、逆に人数が10人を超えチームマネジメントに限界を感じ始める Androidチームを2つに分割しても回る体制を設計。(今年の4月から運用中)	チームマネジメントに興味がある人 チーム拡大を検討している人	25 minutes	日本語 / Japanese	開発体制 (Development Process)	yymsdk
What's new in Android Studio in 2022	みなさん、Android Studioでのアプリ開発を楽しんでいますか？ Android Studioは昨年以上のスピードでアップデートが進み、Android Studio Chipmunk（2021.2.1）をプロダクト開発に投入したと思ったら、2022年7月14日現在、Android Studio Dolphin（2021.3.1）がBeta 5となっているほか、Android Studio Electric Eel（2022.1.1）もCanary 8となっており、次のAndroid Studioもそろそろでてくるんだろうか…？という推測ができる状態になっています。 これらアップデートの要点をつまみ、必要なタイミングで最新のAndroid Studioを投入、移行することは開発において非常に重要な要素となっています。最新版のAndroid Studioやそれに付随するAGP/Gradleの更新点を正しく理解し、便利に利用できる機能を見つけることはチームや個人の生産性や開発速度に貢献するでしょう。 本セッションでは現行のAndroid Studio/AGP/Gradle各バージョンのリリースタイミングやそれぞれのざっくりとしたアップデートの紹介から始まり、Dolphin、Electric Eelでの変更を中心に目玉となりそうな新機能に着目して「Android Studioの潮流」を見ていきます。このセッションが皆様のAndroid Studioのアップデートへのアンテナを張るきっかけとなれば幸いです。 <アジェンダ> - 現在リリースされているAndroid Studio - 最新のAndroid Studio/AGP/Gradleを投入するためにやること - エミュレータ周辺機能の更新ポイント - Logcatの更新ポイント - Jetpack Composeのプレビュー周りの更新ポイント - etc...	- Android Studioを使ってAndroidアプリ開発をされている方 - Android Studioの更新タイミングに迷っている方 - Android Studioの更新で躓いたりした経験がある方	40 minutes	日本語 / Japanese	開発ツール＆サービス (Productivity and Tools, Service)	Yoshihiro Wada

一歩先に行くためのAndroid開発情報キャッチアップ・整理法	Android アプリ開発では、Kotlin / Gradle / Android / Jetpack などの言語・ライブラリに強く依存します。これらのライブラリは頻繁に更新され、日々様々な API が追加・非推奨になっています。特に Jetpack Compose 周辺のライブラリではこの傾向が強く、様々な技術ブログ等にまとめられた情報がすぐに陳腐化することも稀ではありません。また、Android のバージョンアップデートでは、対応しなければユーザの体験を著しく下げてしまうものもあるため、早く正確にキャッチアップすることが求められます。これは特に Android アプリ開発を始めたての開発者にとっては混乱の元です。
	
	この問題を解決するシンプルな方法は、1. 公式のアップデートを絶えず追うこと・2. イシュートラッカー、プルリクエスト を追うこと・3. ドキュメントだけでなくライブラリのコードを読むことの 3 点だとスピーカーは考えています。
	
	本セッションでは、Android アプリ開発に特化した情報の検索法、コードリーディング、整理の方法を紹介し、Android に関しては、自 走して最先端の技術・情報を追うことができるようになることをゴールとします。ここで紹介することは一例にすぎないですが、特にアップデート情報の海に溺れている人を救うことができると思います。
	
	<アジェンダ(仮)>
	- Android / Jetpack の情報の漁り方
	  - Android Developers を正しく読む
	  - target SDK 間での差分を見る
	  - release note を読む
	  - issue tracker 見る
	  - Android CodeSearch をうまく活用する
	  - KDoc / Javadoc を追う
	- Kotlin のアップデート情報のキャッチアップ方法
	  - release note の読み方
	  - You Track を読む
	  - Kotlin/KEEP を読む
	- 必要な情報を自分なりにうまくまとめる
		Androidのライブラリアップデート情報の 多さにうんざりしている人
		Android情報の海におぼれている人
			25 minutes	日本語 / Japanese	その他 (Other)	Kazuki Chigita

なぜ、BaseClassはNGなのか	BaseActivity, BaseFragementなどよくあるBase Class。 9周年を迎えたLineマンガアプリの歴史からたどり着いた「Base ClassはNG」という結論。 コード例や実際の歴史からNGになった理由を説明していきたいと思います。 ### アジェンダ（予定） - 単一責任の原則から考える - 開放・閉鎖原則から考える - 未来に発生するコストから考える - BaseActivityを置き換えていく - BaseFragmentを置き換えていく - まとめ	- レガシーなAndroid Appを良くしていきたい方 - Base Classに関するモヤモヤを晴らしたい方	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	daasuu

その非同期処理, Kotlin Coroutinesでどう書くの？	Androidアプリにおいて、通信処理や、ファイルのReadWrite処理など、非同期処理を実装することは、避けては通れません。 私たちは、AsyncTaskやRxJava。果てはコールバック地獄など、さまざまな方法を用いて非同期処理を実装してきました。 それがCoroutinesを用いることで、コードはシンプルに書けるようになります。 本セッションでは、Coroutinesの簡単な使い方から、実際によく遭遇するユースケースまで、さまざまな具体例を見ていきながら、Coroutinesへの理解を深めます。	非同期処理をシンプルに実装したいけれどもCoroutinesでの書き方がイメージできない方	25 minutes	日本語 / Japanese	Kotlin	sohichiro
よちよちモバイルスクラム〜導入編〜	# 概要 昨今、スクラムという言葉をよく聞くようになりましたね。自分のチームにもスクラムを取り入れたいと思ったことはありませんか？ スクラムを導入するにあたってはなぜスクラムをやるのか？何を持ってスクラムなのか？をまずはしっかりと知る必要があります。 この発表ではリーンXP・ウォーターフォール・メテオフォール（！）とさまざまな開発手法を経験し、現在スクラムを実践している新米スクラムマスターがスクラム導入のために学んだ基本と、Android,iOS,サーバーサイド共存のチームにどうスクラムを当てはめるか、その試行錯誤を紹介します。 一緒にスクラムを学び幸福と成功を手に入れましょう！！ # 発表のゴール * スクラムをしたくなる * スクラムとはなんなのかわかる * スクラムを導入できる気がしてくる # 予定している内容 ## Why スクラム * なぜスクラムをやるのか？ * ウォータフォール辛すぎ問題 * スクラムはなにをもたらす？ ## What スクラム * スクラムとは何か * 定義とマインド * ガイドにないが押さえておきたいこと * Point:幸福は成功 * Point:チームを洗練させると生産性が2000倍になる ## How スクラム * スクラム導入をどう進めるか（私たちの場合） * チームへのマインド浸透 * ソフトな実施 ## あるスクラムチームの場合 チームに導入しているスクラムのルール・ツール・運用方法について紹介します。 * ルール * POとDevelopers * スクラムイベント * エピックとストーリー, そしてタスク * Point:ストーリーはどう切るのか？ * Android,iOS,サーバーサイドはどう協調する？ * ポーカー見積 * 進ち指標 * Point:幸福度を測る * ツール * Zenhubのあれそれ * 現状と課題とこれから ## まとめ	* スクラムに興味がある方 * ウォーターフォールが辛い方 * 幸せになりたい方	40 minutes	日本語 / Japanese	開発体制 (Development Process)	Saiki Iijima
9周年を迎えたアプリのModernize戦略	Lineマンガは9周年を迎えました。 9年前に書かれたコードもまだ現役で動いていたりします。 その様な状況の中、リファクタ戦略を描き、順番に実行していき、 より生産的な開発, コードになる取り組みを行なっています。 ## アジェンダ（予定） - CD, CIの取り組み - lint実行をdangerで通知。 - Base ClassがなぜNGかの歴史を持った考察 - BaseActivity, BaseFragmentの解体方法 - 単一責任の法則、解放・閉鎖原則から実践する - 「Legacyモジュール」を作れない場合のマルチモジュールの取り組み - Application classへのstatic な依存が多く、一度に大きなモジュールを切り離せない場合のマルチモジュール化 - Dagger Hiltの導入へ - ビルド高速化戦略 - Jetpack Compose導入 - Previewが遅い問題に対応したモジュール戦略	- 歴史のあるAndroidアプリを改善していくことに興味がある方 - リファクタリングのTipsを知りたい方	40 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	daasuu
不正なアプリを炙り出す新API！Play Integrity APIへの移行	Play Integrity APIは、不正な操作からアプリを保護するために、新しく用意されたAPIです。 例えば、正規な状態でないデバイスや改変アプリからのアクセスを判断したり、ライセンスの付与状況を確認できます。 なんとなく聞いたことがある機能かと思われるかもしれません。 実際その通りで、Play Integrity APIは、以前から提供されているSafetyNet Attestation APIとApp Licensing APIの後継となるAPIなのです。 すでに、SafetyNet Attestatiopn APIは段階的な廃止スケジュールが提示されています。(2023年6月移行期限、2024年6月完全廃止) 本セッションで、過去のAPIと共通している部分や移行にあたって考慮が必要な部分を確認し、今から移行に向けて備えましょう。 ■セッション構成(予定) ・Play Integrity APIとは ・旧バージョンとなるSafetyNet Attestation APIとApp Licensing APIと変わったところ/変わらないところ ・移行スケジュールの考え方 ・移行のために考慮しなくてはならないこと。 - 処理構成の変化 - 結果の対応関係 - 注意するべきオプションや例外的な状況など 上記に加え、当日までに、root検知回避ツールなどにより、考慮すべき内容が増える状況が生じた場合には、その内容についても解説を行います。	・すでにアプリでSafetyNet Attestation APIとApp Licensing APIを導入している開発者 ・これからアプリ保護のためのAPI導入を検討されている開発者	25 minutes	日本語 / Japanese	Security / Identity / Privacy	Akihiro Shiota
自分の手で施策を回す	みなさまの開発されているプロダクトでは、1つの施策を立案、実行し、その振り返りを行うためにどのようなフローを通して実装されていますでしょうか？ PdM/PjMのような立場の方から実装のための要件がすでにある程度固まった状態で受け取るようなトップダウンでのフローで開発をされている方もいらっしゃるでしょう。プロジェクトの中で役職の関でなく、メンバー全員で施策の立案から振り返りまでを追いかけるという方もいらっしゃると思います。 では、エンジニアとして「1つの施策を立案から振り返りまでに携わる」には、どのような動きをしていくのがよりプロダクトの成果を押し上げることにつながるのでしょうか？ 本セッションでは施策を起案してから継続の判断をするまでの一連の流れについて事例をもとに紹介していきます。 現在考えているセッションの内容は次のとおりです。 - 施策の目的を定める - どんなサービス体験を考えるか - どんな目的を達成したいのかを定める - 数値目標を決める - 施策のアイデアを出す - ユーザーから意見をヒアリングする - 他のサービスを調査し学ぶ - アイデアを具体的な形に落とし込む - UI/UXを調査する - 施策のスコープを考える - どのくらいの期間でどのくらいのものを出すか - NativeかWebViewか - 成し遂げるデザインとその妥協 - 関係するチーム - アプリで完結するか - iOS / Androidの展開 - 他に提供しているサービスとどう関係するか - 施策を効果測定する - 施策の続行と撤退について - 次の施策へいかす 施策はサービスの方向性を決める重要なもので、自身がコミットしているアプリ・サービスをより俯瞰した視座で捉えることが求められます。 本セッションが、意識的に開発や施策を行うためのフローを作り上げ、より良いプロダクト・開発体験を生み出すためのきっかけとなることができれば幸いです。	- 施策の遂行に関わる全てのAndroidアプリエンジニア - エンジニアと協力し、アプリやサービスをより良くするミッションを持っている方 - 関わっているアプリやサービスの成果を押し上げたい、と考えている方	40 minutes	日本語 / Japanese	開発体制 (Development Process)	Ryo Yamazaki, Yoshihiro Wada
今日からできる！Jetpack Composeカスタムレイアウト	Jetpack Composeは、標準のレイアウトを組み合わせることで様々なレイアウトを組むことができます。一方、実際に開発をしていく中で標準レイアウトだけでは実現が困難なレイアウトを実装する機会もあるでしょう。 Jetpack Composeでは、そのような場合カスタムレイアウトを活用することができます。 標準レイアウトと比較してより自由度が高いレイアウト配置が可能になり、カスタムレイアウトを使いこなせるとアプリ上で表現できるデザインの幅が広がります。 本セッションでは、カスタムレイアウトの基礎知識から複雑なレイアウトの実装までを実例を交えながら解説したいと思います。 想定する内容 ・Jetpack Compose カスタムレイアウトについての基礎知識 ・Jetpack Compose カスタムレイアウトによるレイアウト実装方法 ・Jetpack Compose カスタムレイアウトによるレイアウト実装の実例	・Jetpack Composeの導入を検討している方 ・Jetpack Composeで複雑なレイアウトを実装する方法を知りたい方	25 minutes	日本語 / Japanese	Jetpack Compose	Horie1024

モバイルアプリのユーザープライバシー新時代	近年、アプリ開発においてもユーザープライバシーの重要性が増してきています。セッションではモバイルアプリでのユーザープライバシーへの配慮方法を紹介し、実装の指針を紹介します。 基礎的な知識から実装方法、ユーザーフレンドリーな仕様をカバー対象とし、将来導入されるPrivacy Sandboxといった新しいアプローチへの理解を推進することが目標です。 - パーミッションの制限、ロケーションの適切な取り扱い方法 - センシティブデータの管理・推奨手法 - デバイス識別子や行動分析などでのユーザートラッキングの注意点 - プライバシーに関するリスクを最小限するプラクティス たとえばパーミッションでは2022年1-3月にかけて「使用していないアプリの権限を削除」する挙動に変わっています。 これはユーザーのプライバシーを保護するための変更ですが、日頃からアップデートを追いかけていないと気づくのは難しいでしょう。 さらにプライバシーに対する理解度は組織、チーム、個人でさまざまなのが現状です。 プライバシーを意識した機能、その導入の背景、ユーザーにとって嬉しい仕様を理解してチームで議論をリードできると強みになるはずです。この機会に知識を更新しましょう！	- プライバシーに関心がある方 - プライバシーは大事だと感じつつも、詳細をまだ調べきれていない方 - ユーザー満足度の高いアプリにしたい方 - Google Play Storeから不意にプライバシー違反で怒られたくない方	25 minutes	日本語 / Japanese	Security / Identity / Privacy	mhidaka
音声配信アプリにおけるAndroidを使った音声配信の全てと裏側	トーク概要 - Voicyでは日々パーソナリティさんという音声投稿者による音声配信がされております。 - Androidにおいて音声配信をする上で欠かせないMediaPlayer / MediaRecorderを利用しております。 - MediaPlayer / MediaRecorderには多種多様な機能がありますが、実際サービスを扱っていく上で一体どんなことに注意し何を必要があるのか？がわからないことが多いです。 - 本トークでは音声配信をAndroidで実装する上でどのように考え実装するべきなのかを全てと裏側という形でお伝えしたいと思います - MediaPlayer / MediaRecorderを利用した音声の取り扱い - 音声収録 - 音声配信 - etc - Voicyで利用している音声配信の提供方法 - 収録配信 - リアルタイム配信 - 課題や今後の展望	音声配信のAndroidアプリを作成している方 / Androidのコア機能を利用してアプリ	25 minutes	日本語 / Japanese	保守・運用・テスト (Maintenance, Operations, and Testing)	entaku
「OK, Google」でアプリの機能を呼び出す	Androidスマートフォンに標準搭載されているGoogleアシスタントでは「Ok, Google」のキーワードで呼びかけることで、Googleでのキーワード検索、アラームの設定といったGoogle/Androidが提供している機能を手軽に操作できることが有名です。しかし実は、我々が開発するアプリもGoogleアシスタントに対応させることができます。 2021年のGoogle I/Oでも、アプリをGoogleアシスタントに対応させるための「App Action」を実装するAPIについてより詳しく発表されました。 音声操作を実装するためには、ユーザーが発話する自然言語を分析し、微妙な言語の「ゆれ」を吸収する（例えば「検索する」と「探す」を同一操作とする）ことで「どの機能を呼び出したいのか」を明らかにする処理が必要です。こういった自然言語の分析を手軽に実装できるAPIが提供されています。 Googleのドキュメントを確認すると対応言語に日本語は含まれていないため一見使えないように思われますが、開発してみると日本語でも実際に動作させることが可能でした。 今回はこの音声操作を実現する「App Action」の実装や動作確認の方法についてお話ししつつ、ドキュメントでは扱われていない日本語での操作などの程度実現できるのかについてお話しします。	・Androidアプリの音声操作や音声UXに興味がある方 ・開発しているアプリの機能をユーザーが手軽に呼び出す方法（ハンズフリーでの操作やショートカット機能）を検討している方 ・日本で馴染みのない音声操作をユーザーに活用してもらう（機能仕様を促進する）方法を探っている方	25 minutes	日本語 / Japanese	Androidプラットフォーム (Android Platform)	horitamon
The Clean Architecture in Android ～「く」の字から実装まで～	クリーンアーキテクチャはUncle Bobにより提唱された、より開発・保守・運用のしやすいソフトウェアアプリケーションを設計するためのアイディアの一つです。 一枚の記事で説明されるシンプルなアイディアですが、そのシンプルさ故に広い解釈があり、完全に理解するのは困難を極めます。 しかし、どんなアーキテクチャも正しく理解し考えながら利用しなければ真の効果を発揮できません。それどころか単なる窮屈な制約と感じられてしまうでしょう。 本発表ではThe Clean Architecture の記事と書籍を元に、そもそもクリーンアーキテクチャとは何か？から始め、利点を知り理解を深めます。 複雑化するアプリをより堅牢に設計するためのよい足がかりとなること間違いなしです。 ##### 予定している内容 # 理論編 * 概要と立ち位置 　* 実装の方法ではない * MVVMやMVPとの比較に意味はある？ * 余談：iOSにおけるクリーンアーキテクチャ * 更に詳しく 　* 産み出される五つの特徴 * ただ一つの大事なルール 　* 四つの層 　　* 例えばどこに何を書くか 　　* UseCaseについて 　* 境界のまたぎ方 # 実装編 ## 例えばこんな実装（とあるKMMを使った大規模アプリの場合） * レイヤーとクラス * プレゼンテーション層をちょっと詳しく * 効果 ## まとめ	* クリーンアーキテクチャいまだによくわからんな方 * そろそろクリーンアーキテクチャ知っとこっな方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	Saiki Iijima
ネイティブアプリにWebViewをうまく溶け込ませる実装のTIPS	アプリでWebViewを使う、と聞いてなんとなくいや～な気持ちになったりしていませんか？ ネイティブアプリ開発に慣れているとWebViewを使うとネイティブアプリっぽさが薄れてしまう、WebView周りのアーキテクチャで困る、などWebViewを避ける理由が多くなってしまうかもしれません。 しかしWebViewは使い方や付き合い方を間違わなければネイティブアプリでは難しいことが簡単に実現できることもあり、強力な武器になります。 本セッションではネイティブアプリの中でWebViewを使うことを選択肢として認識してもらえるように、WebViewとの付き合い方で意識していること、実装の際に気をつけていることなどをお話しします。WebViewをアプリ実装のメインには据えず、あくまでもViewの一種としてうまく利用する方法になります。 ・WebViewをネイティブのコードに溶け込ませる設計（JavaScriptとどう責務を分担するか） ・WebViewをネイティブの画面に馴染ませる見せ方（体験設計について） ・AndroidのWebViewとiOSのWKWebViewの違いと固有の問題 ・WebViewの挙動で気をつけるべき点（スクロール・クリック・長押し・リンク・選択など） ・開発する際のWebViewのデバッグのやり方 ・WebViewが向いていること・向いていないこと	・ネイティブアプリでWebViewを使うのは何となく嫌だなあと思っている方 ・WebViewを使ったアプリ開発の事例に興味のある方	25 minutes	日本語 / Japanese	アプリアーキテクチャ (Application Architecture)	六々 (@496_)
コルーチンを使って処理の見通しをよくする - リアルタイム放送の品質を向上し、保つために	私が現在開発している「リアルタイム音声放送」では、アプリとサーバー間の通信だけでなく、同時にアプリ同士のP2Pの通信も行っており、複数の通信状態を非同期に、かつ並行処理も行いながら管理しています。 開発が進むにあたって機能を追加/修正してコード量が増加していくと、こういった非同期処理や並行処理が複雑に絡み合うようになっていきます。こうしたコードは可読性やメンテナンス性が下がり、本来持続しなければいけない処理が待たれずに次の処理へ進んでしまったり、受け取れないはずの操作が状態管理の不備によって受け付けられてしまったりと、難解な不具合を混入させやすい状態になります。 今回はこういった課題を解決していく手法としてのコルーチンの活用方法について、プロジェクトの品質向上の視点からお話しします。	・コルーチン自体は知っているが、どういった場面で有効なのかより具体例を知りたい方 ・非同期処理や並行処理での状態管理に悩まされている方	25 minutes	日本語 / Japanese	Kotlin	horitamon
Graphics for Android Developers	In this session, we will describe fundamental and important graphics concepts, as well as how they are used in the Android APIs to render your applications. You will learn how to build exciting and efficient visual effects to improve user engagement. The concepts presented can be applied to both Jetpack Compose and Android Views.	Basic Android app development	40 minutes	English	UI・UX・デザイン (UI・UX・Design)	Romain Guy, Chet Haase
Why Projects Succeed: Lessons Learned from the Android OS	The story of how Android came to be, and how it managed to survive and thrive in a crowded field of competitors, is an interesting story of people, teams, and vision. But it's also an educational tale of how products can succeed, which can help inform future ideas, startups, companies, acquisitions, and projects throughout tech and business overall. This session will draw from the recently published book, Androids: the Team That Built the Android Operating System, to see what we can all learn from the Android project, which started as two people building a camera OS and resulted in a platform running on more than 3 billion devices today.	No prerequisite knowledge of Android	40 minutes	English	その他 (Other)	Chet Haase, Romain Guy
What's New in Android Developer Tools	In this talk, the engineering director for Android Studio will go through recent changes to the IDE and build tools, highlighting important changes and demoing some interesting new features.	None	25 minutes	English	開発ツール＆サービス (Productivity and Tools, Service)	Tor Norbye