WHITE PAPER

android tv

Martin Sonc

Beenius
www.beenius.tv

TABLE OF CONTENTS

INTRODUCTION

Congratulations on your decision to update your knowledge.

Enjoy reading our white paper "Android TV", written by our amazing colleague ***Martin Sonc*** *– Android Developer.*

Feel free to e-mail us your feedback: marketing@beenius.tv

THE WHO?

TV Operators

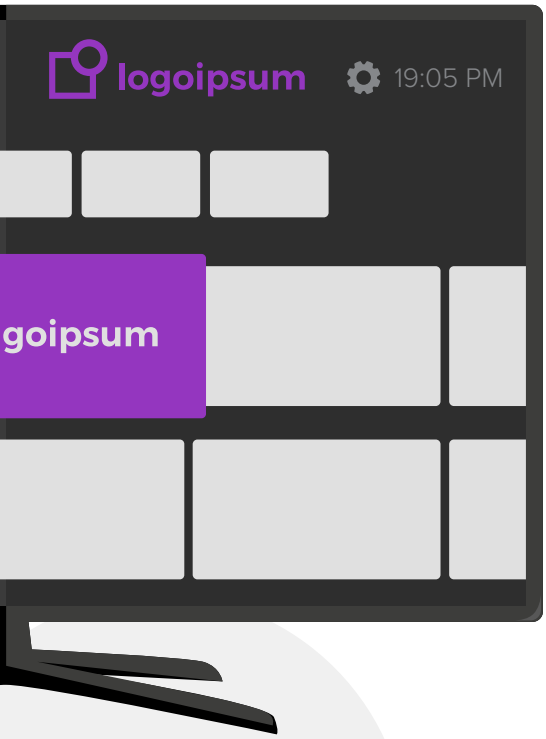
It all starts and ends with the operator, which also makes it a partner that must answer some difficult questions, mostly stemming from the way the users viewing habits are, slowly but certainly, evolving. Taking that into account there are multiple ways for the operators to approach **AndroidTV**, and it all starts with a launcher. **Beenius Launcher**.

A launcher in Android world is just another word for the home screen which is mandatory for **phones, tablets, TVs, STBs** and everything in between. Under the hood though, it is just another application and the operators can decide to modify, augment or even replace it. In the best-case scenario, one solution could (and should!) cover all of the cases below, which is exactly how we approached the development of the Beenius Launcher:

Non-modified AndroidTV with a custom application

Nowadays AndroidTV comes preinstalled on a wide range of TVs and TV dongles, enabling a great living room experience without an STB and providing an amazing opportunity for the TV operator to lower their cost since there's no need to deal with hardware, the operator simply provides an application to their users. Beenius Launcher takes advantage of multiple customization options the platform provides, such as exposing the data to the **Google Assistant**, allowing for voice control, providing curated and

personalized recommendation rows the user can add to their home screen and even integrate with third-party applications through deals with TV networks such as **HBO, Netflix, FOX**, and others.



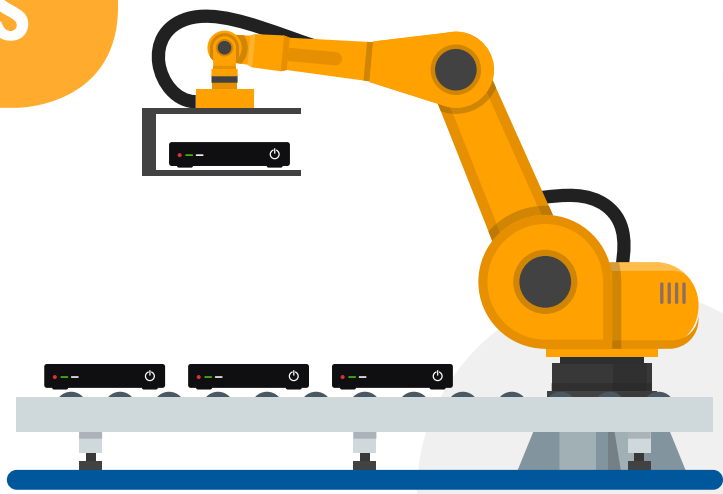
AndroidTV with a branded launcher and a custom application

In case the operator wants or needs to deploy an STB to their users it can decide to do so without having to create their own launcher. The default implementation, provided by **Google**, allows for multiple types of customization while still providing a great and open user experience. With Beenius Launcher, the operator can augment the launcher by replacing certain **icons**, modify the **color scheme**, pre-populate the home screen with **rows** of content and even prevent the user from removing some of them, while still accommodating all the functionality referenced.

Custom build of AndroidTV with a custom launcher

In this scenario, the default **AndroidTV launcher** would be replaced by the **Beenius Launcher** and distributed as a part of the firmware, which in term means it has to be certified by Google (or one of their certification partners) and must follow their user experience guidelines. Beenius Launcher gives the operator complete control over the box, including potentially limiting access to external applications. One of the important parts of the user experience is onboarding or provisioning, which can actually be performed as an additional step during the device setup. It allows for more freedom but comes with additional cost, particularly in terms of certification and maintenance.

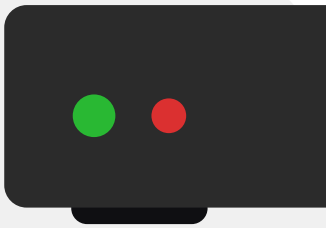
STB Manufacturers



If the operator opts to deploy their own hardware, partnering with the right **STB manufacturer** is crucial, especially since AndroidTV requires a long-term commitment to provide mandatory updates during the first couple of years of the operation. The list of responsibilities is by no means exhaustive but should at least provide a rough overview and explain the decisions required to make the deployment a success.

Beenius as a **media system integrator**, together with our trusted **STB providers**, has over a decade of experience in consulting and guiding operators throughout the whole process of choosing the most suitable hardware.

STB & Remote



It might sound trivial, but it really (really!) is not. With the average life span of an STB being around six years, any decision made at this point can come and haunt you for years to come. Google requires any AndroidTV STB partnership to be backed by a long-term agreement, providing a minimum of two years of security patches and operating system updates. It is hard to predict what requirements future media formats are going to bring, making it hard to find a good compromise between hardware specifications and cost.

Customization

The manufacturer oversees the firmware so, since the launcher application is baked into it, that makes them the main distributor of it. But the launcher is not the only modification that can be done on the STB, by far. The TV still is the centerpiece of the living room, making it an ideal candidate for it becoming an IoT hub or a gaming station without the console with the rise of platforms like **Stadia** and could even be an entry point to the world of **VR**.

Certification

Even though the certification can be a cumbersome and time-consuming process we prefer to look at it as more of a blessing than a curse. It is basically a long array of tests Google and its partners perform on the hardware and software side to make sure the operating system works as expected, meaning the app developers don't have to deal with certain deviations in the implementation as it tends to happen in **AOSP projects** where there is no oversight. It can also simplify and speed up certain third-party certifications, such as the ones required by **Netflix** or **Amazon Prime Video**.

Application Developers

From an application developer standpoint following the guidelines and dealing with the certification processes can feel a bit overwhelming. As mentioned before, the certification process offers a lot more stability and predictability throughout the development process while the issues can be mitigated through careful planning and, obviously, by picking the right partners for the job. Due to the extensive knowledge of the **Android platform**, and the particularities of AndroidTV, **Beenius is a proven and competent partner** in the application development and certification process.

Google

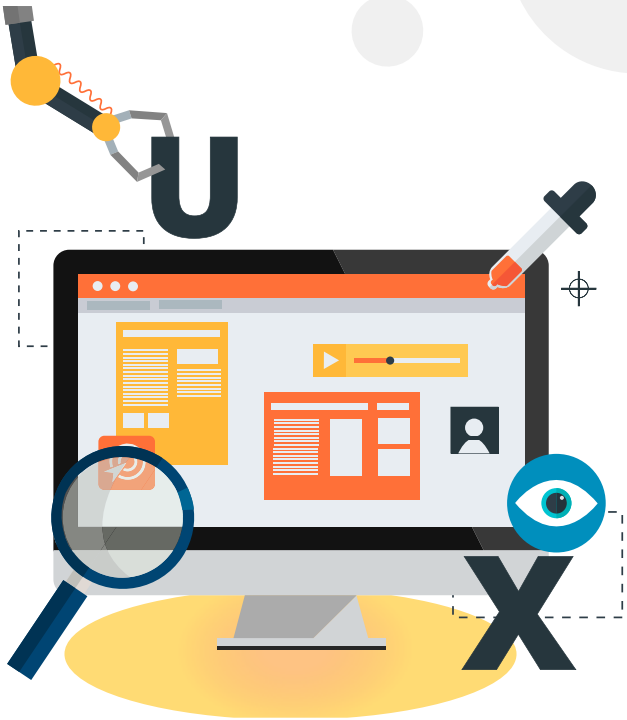
Last but not least, Google is a partner that you do not get to choose and acts as a **certification authority** - a gatekeeper between the user and mediocre hardware and software. When starting a project a **technical account manager** (also known as a TAM) gets assigned to it, a single point of contact that should be able to provide guidance and mentoring throughout the process.

THE WHAT?

At this point, the playing field is set, the roster is selected, and the playing positions are allocated, so it is time to look at what exactly AndroidTV actually is.

In the broadest sense, it's just a fork of Android with some slight modifications to support specific broadcasting functionalities. But, it's much more than that and should not be treated just as a simple operating system but an ecosystem consisting of an operating system, development libraries, guides and documentation, **Google Play Services** and, most importantly, it comes with the **Google Play Store** and its constellation of different applications. Plus, it acts as a **Chromecast** device!

User Experience



As mentioned before, with the Beenius Launcher, the operator can decide to customize the user experience on the device in accordance with the guidelines provided by the AndroidTV team. So let's look at the design language Google is promoting as their signature **"10-foot experience"**.

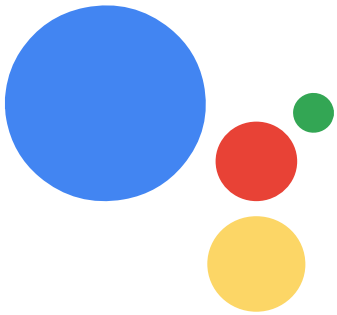
The primary screen is made from horizontal rows of content called **channels**, containing anything from applications or video content. The installed applications prepare the content, include one row to the home screen by default and expose even more to the users, that then decide if

they want to add more to the home screen. The rows can easily be added, rearranged or removed and the content should, as a rule, be only a click away.

Channels support all sorts of functionality, including previews, rich metadata, custom icons and much more, and is a great way for the app developers to show off the content and lure the users into their application.

The navigation is done primarily with the **directional pad** on the remote or by **voice commands** (a Bluetooth remote with a microphone is actually a requirement). One of the major issues of the platform is text input, using the **on-screen keyboard** is an absolute nightmare, so the app developers should do whatever it takes to avoid it and use **voice for search** and a mobile device for user authentication.

Google Assistant



How do you search for “Monty Python’s Flying Circus” when the keyboard is borderline useless? Well, just talk to your remote! The **Google Assistant integration** on AndroidTV is amazing (provided the remote control was properly designed) and allows two tiers of integration, both supported by **Beenius’ middleware** and the **Beenius Launcher**.

The simplest option is on-device search, where the application exposes data in a predefined format for the Google Assistant to search through. The documentation is great, and the implementation is really straight forward. It works but can be a bit limiting when trying to add some advanced functionality, such as register additional keywords.

Fortunately, Google saw this problem coming and is now offering a special integration path for such advanced use cases, called **cloud-to-cloud integration**. In this scenario, the entire **VOD library** is exposed to Google through an integration API, which allows for a way better user experience, advanced (contextual) actions and even custom routines, such as **“remove all recorded content”**.

WHEN?

History Lesson

Using the **Android operating system** as a foundation for a new and improved TV experience is definitely not a new idea and has a quite interesting history, so it makes sense to look back before kickstarting our journey into the future.

Google TV

Google TV, Google's first foray into the TV business, was initially released in 2010 with hardware provided by Sony and Logitech and received a lukewarm reception on both the business and the consumer side.

> *Google TV feels like an incomplete jumble of good ideas only half-realized, an unoptimized box of possibility that suffers under the weight of its own ambition.*

It is honestly hard to disagree with that review, the only functionality that seemed to be worked out was the search, and even that was half baked. The entire platform ended up being more or less just a convenient way to launch the browser with **Flash support**. The damn thing shipped with a wireless mouse and keyboard!

Unsurprisingly, the platform wasn't doing well, and it soon started to look like that was one of those products Google uses to test the market and was soon to be replaced with Android TV.

Open Platform Solution

Meanwhile, looking at the mess that was Google TV, Beenius already started working on bringing Android to the STBs. Working closely with the hardware manufacturers and content providers we demonstrated the **first build** in 2012. After the initial demo we spent years addressing the security and stability issues and officially released our platform in 2013. In 2015 we released the brand-new client applications and released a **first Android DVB Hybrid solution** in 2017.

androidtv

It seems like Google soon realized Google TV in its entirety wasn't going to work and decided to scrap the project and replace it with a much more polished Android TV, announced at the **Google IO conference** in 2014. The users finally got the platform they expected, a polished interface backed by the Android ecosystem, including **Google Play Store**, improved search integration, Chromecast functionality. One of the interesting decisions was to integrate **Google Play Games** from the beginning, enabling the users to use the device as a gaming platform.

Beenius meets Android TV

Being ahead of the curve is always a blessing and a curse. So even though Beenius spent years convincing operators, content providers, and clients that Android is the way to go the path to where we are now was a long and interesting journey, worthy of a standalone article.

Not to go into too many details, we soon had plenty of clients and **Android AOSP STBs** deployed all around the world (25 countries and counting!), all while keeping a close eye on the **Android TV project**. When the platform seemed mature enough we were ready to go - the engineers were there (Android is in our DNA anyway), we have extensive knowledge of the underlying system so building an AndroidTV experience is a natural step forward.

WHERE?

There is plenty of areas where Android TV can make a huge impact, augmenting and improving the customer experience while alleviating many of the pain points operators have to tackle.

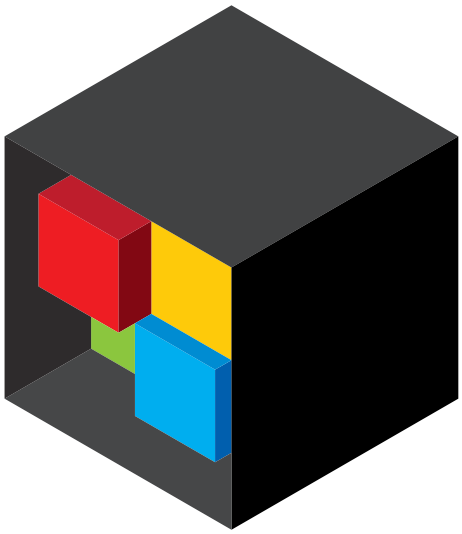
Living Room

As mentioned before, the TV is still a centerpiece of any **modern living room** - otherwise, where would you point all the furniture? And, while everything around us was getting smarter and more and more powerful, the **TV** still seems like the most underutilized piece of hardware everyone has and loves. It seems bizarre that with the huge leaps forward in the user experience on literally every other platform the screen we seem to use and enjoy the most was lagging behind.

Understandably though, with huge deployments and tough technical challenges of getting the video to the user, everchanging standards and exponential growth in video fidelity, the focus was primarily on the technical aspect. What good is a great user experience if the video is broken? Same as on the web or in the mobile application development, the scope started expanding and we finally are in a position where we can put the user front and center, we now know what they want and have years of **user data and analytics** to guide our path forward. We learned that the best interface

is no interface, so we're bringing search to the foreground, improving the discoverability with an advanced recommendation engine and focusing hard on reducing the cost through targeted advertising. We are putting a lot of effort into user research through **in-person sessions, extensive analytics**, and **A/B testing**.

Content Provider



Fragmentation is a bad thing. It seems like Google recognized that early on, so Android TV offers quite a few mechanisms to help with the distribution and management of the applications while improving the reliability of the platform, with some great improvements scheduled for the near future. There are plenty of tools that can come handy, especially when distributing the applications, which is what we will focus on in this section.

The **Google Play Store** offers a great set of tools for getting the right application to the right consumer reliably and in a timely manner, from the testing phase onwards, even for the **Operator Tier launcher** implementations. One of the major drawbacks, however, is that the user needs to have a Google account associated with the device - though, there are some great improvements in the works, which will allow certain updates to circumvent that limitation.

One of the first issues the operators are facing when preparing for deployment is testing, and fortunately, there are some great options available out of the box, the most important being support for **pre-release testing**. There are three release tracks available:

Alpha Track

Should contain the most bleeding-edge builds of the application. At Beenius we recommend the clients to publish to the alpha track on the weekly, if not daily basis. While the application should work, there are no guarantees it will work reliably and the track should only be used to distribute it to internal testers. It's also a great place for the final users that are technically savvy enough to have a sneak peek of what's to come.

Beta Track

The applications should only be promoted to the beta track when the majority of bugs have been resolved, and the application is close to being complete. It's a great way to show off the application to a limited set of users and collect feedback. There are no better testers than the final users!

Release Track

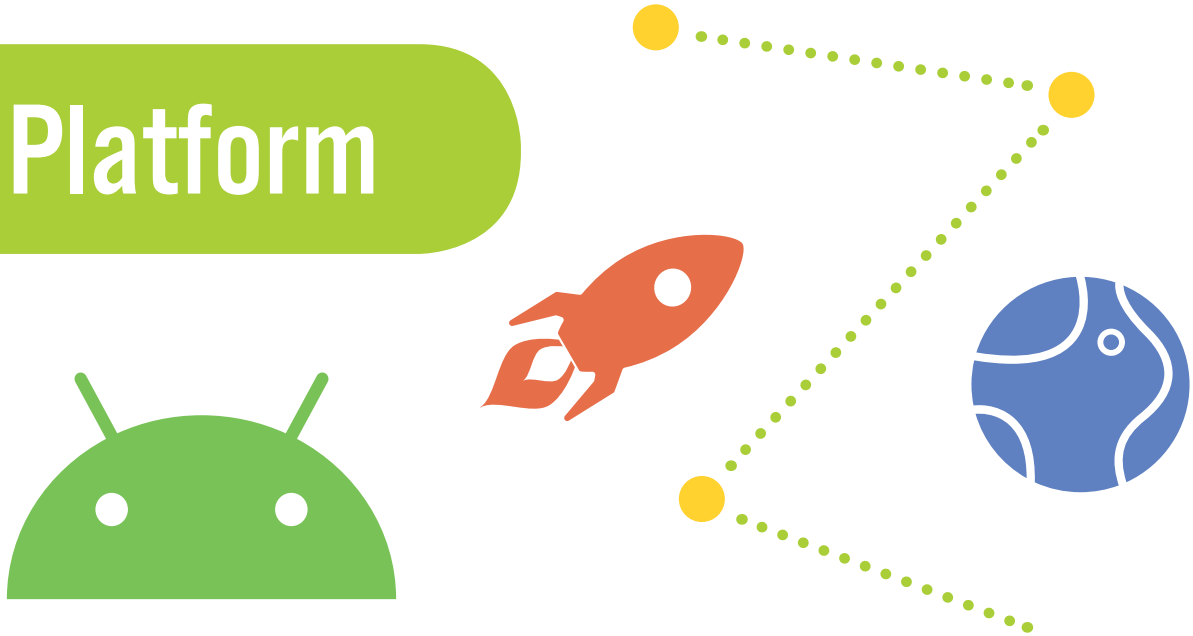
Contains the public release, where the application is delivered to all the users.

Each offers two enrollment options, a public one where the users can sign up by visiting a dedicated website and closed testing, where the operator manually adds the friendly users that will be testing the application. We employ a highly automated continuous integration system that handles publishing on different tracks automatically, with no developer intervention.

WHY?

We understand that transitioning to an entirely new system can seem like a risky move, especially if talking about an existing deployment, where introducing a new platform can drastically increase the complexity of the entire system. We do however firmly believe making such a transition is worth the effort, especially, but not exclusively, because of the reasons outlined in this section.

Platform



Android is an open-source Linux based operating system that is deployed worldwide and used by more than **2.5 billion** users. And, it used to be a complete mess, plagued with fragmentation, performance and security issues. Fortunately, it seems like those times are over, especially when taking a closer look at a few of the recent initiatives, such as project treble, where the vendor implementation became separate from the operating system. They achieved that by improving and formalizing the **HAL** (hardware abstraction layer), an interface that connects the software and the operating system with the drivers and other vendor-specific code. What that means is that an **OS update** does not require a re-work or even an update on the firmware side.

Even with project treble, the upgrade cycle for Android OS is still way too long and developers need to make sure the applications can support older versions of Android while still following the latest trends and implement features only available in new releases. To help solve that issue the Android team maintains a set of **Support Libraries** that provide support for newer features on earlier versions of Android or gracefully fall back to equivalent functionality.

Other important changes in Android development is the introduction of a new programming language called **Kotlin**, a bigger focus on the software architecture and regular improvements of the IDE and the included debugging and profiling tools.

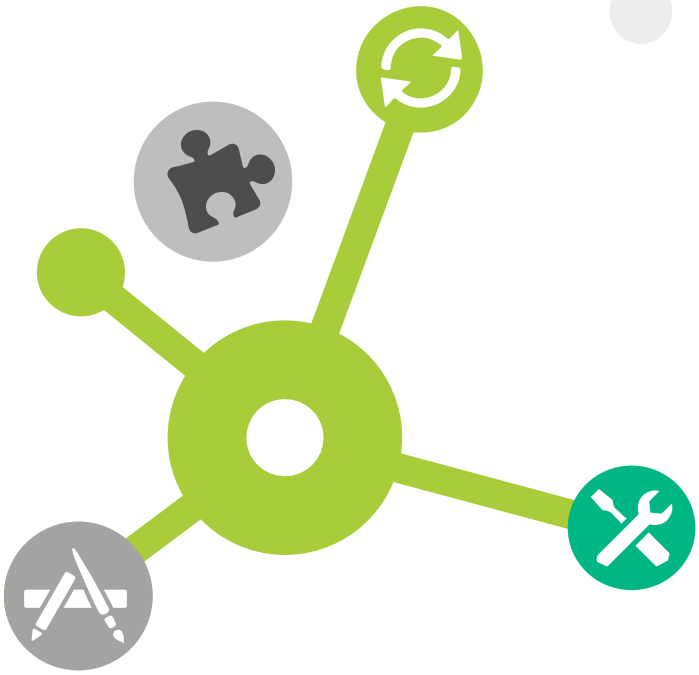
Reliability and Speed

In order to provide an application that offers a great reliable viewing experience, we need to ensure both the operating system and the application is as reliable as possible. We'll talk more about how to achieve and measure that on the application side, but it's worth mentioning a very important thing right at the beginning: Android is, as said before, a mature platform which means there's already quite a few really good developers with years of experience at your disposal. From personal experience, developing for the TV or mobile devices is not that much different, so "up-skilling" to a TV developer should come naturally to most experienced Android developers. You might say that doesn't directly impact the reliability and speed aspect, but I beg to differ - programming is more than just producing quality code, but also about being able to use the tools, and there's plenty of them at our disposal. Being able to constantly keep an eye on the performance, measure network traffic and identify memory leaks as seamlessly and simply as we do in Android world is something that'd be hard to find in any other **TV application development framework**.

Familiarity

Since Android and Android TV follow the same **UI** (User Interface) and **UX** (User Experience) principles, the Android users should, as long as your application follows those same guidelines (which it absolutely should!), have no problem with onboarding and navigating the shiny new app. In Beenius we decided to follow the general design language of the default **Android TV launcher** and the **Material Design guidelines** while working hard to identify the places where we can improve the interface without sacrificing the familiarity. We want the user to be able to feel at home the moment they launch the application for the very first time!

Ecosystem



As mentioned a couple of times before, Android TV doesn't only borrow the underlying code from the Android OS but also adopts and embraces the entire ecosystem of applications, with all the large players having native Android TV applications on the **Google Play Store**. And, we at Beenius decided to play nice and make sure the users can access and install any of the applications available on the market. People are going to watch **YouTube** and **Netflix** if we want it or not - we might as well enable them to do so through our own application where we can, with the aid of our great **recommendation engine**, offer the content users can enjoy as a part of their TV package or as a one time purchase through our **VOD library** and **purchasing system**.

HOW?

Goals, ideas, and principles are a great thing to have, but in the end it's all about the execution. How do we create a TV experience that's reliable, pleasant to use easy to maintain?

Tools and Best Practices

Application Development

We like to pride ourselves with following the latest standards in the application development world, including the usage of clean architecture supported by the **MVVM model** and packaged using application modules.

Clean Architecture and the MVVM model

It's only recently the architecture debate came into the forefront in Android development, but the basic principles have been the same and well understood for ages. What we're basically saying is that before the code can be reliable and testable it needs to be well organized. The MVVM model breaks down the structure of the application (or in our case a module) into a couple of distinct layers:

View

A layer controlling a part of the user interface (usually a Fragment). Doesn't contain any logic or much functionality at all, but should act as a glue between the view described in the **XML** and the content and behavioral changes requested by the **View Model**.

View Model

Should contain all the logic required to support the behavior and data exposed through a **View**. It not only exposes the data and makes sure it's properly preserved through state changes, but it also reacts to the user and system interactions.

Model

In charge of retrieving and sending data and contains most of the complex logic responsible for presenting the data to the user, as well as handling and responding to user interaction.

The underlying layers are in our case provided as a separate library (or, as we like to call it, an **SDK**), that handles all the authentication, data sending & retrieval, provides an off-line cache, basically acts as a foundation on which the new generation of Android-based Beenius applications are built.

Application modules

We have further decided to split the application into distinct modules, allowing for greater flexibility and empowering some next-generation customization options. Each module can be independently developed and tested, dynamically enabled or disabled and in certain cases even replaced remotely using the **Dynamic Feature Module** functionality.

Remote Error Collecting

We try to encourage our clients to allow us to collect anonymized user statistics and collect the error data remotely, with no user interaction needed, meaning the first error reports should reach our developers directly and can provide a great mechanism for quickly resolving a great majority of issues the clients are experiencing.

Analytics

The best way to build great software is to listen to the users. Instead of bloating the application with all the imaginable functionality we prefer **gradual increments** and tweaks based on the actual usage of the application, which can differ from market to market and can be deployed as such thanks to the architectural approaches outlined in the previous sections. When considering a larger change in the functionality or even a new module we can roll out the functionality gradually, to a limited subset of users and collect the feedback before releasing to the entire user base. Making decisions like that is only possible if based on real measurements, which in term makes getting the analytics right one of the greatest and the most important challenges in modern application development. By using a combination of **in-house tools** and **Firebase integration** on multiple levels we think we have found a sweet spot in the analytics gathering and processing.