Case study

Building a Complex Parental Control App for Android

Our client is a software provider that has been offering internet protection and parental control solutions for desktop browsers. Given the increasing use of mobile devices by children, our client decided to make a parental control app for Android tablets and smartphones.

The Apriorit mobile development team researched different scenarios for implementing the parental control application and came up with a flexible solution that meets all of our client's needs.

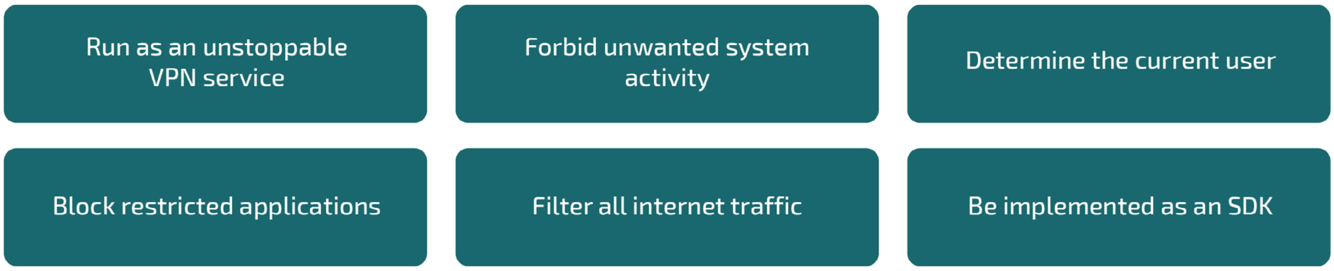
The client

Our client is an internet protection software provider that offers solutions for filtering internet traffic and limiting or blocking access to suspicious or restricted content. These solutions are mostly used for establishing parental control over internet content consumed by children. The key goals of these solutions are ensuring the safe use of the internet by kids and letting their parents decide what types of content they can interact with and to what extent when surfing the web.

The challenge

Our client already had a traffic filtering solution for desktop browsers and wanted to create a parental control app for Android mobile devices. According to their requirements, the final product was supposed to:

CLIENT'S REQUIREMENTS FOR AN ANDROID PARENTAL CONTROL APP



Run as an unstoppable VPN service

Forbid unwanted system activity

Determine the current user

Block restricted applications

Filter all internet traffic

Be implemented as an SDK

www.apriorit.com

When looking for a team that knew how to build a parental control app, our client settled on Apriorit. They delegated to us all research and development activities for this project.

Our solution

For this project, Apriorit formed a dedicated team of experienced mobile developers, quality assurance specialists, and a project manager. Initially, we focused on two primary goals:

- Develop an Android parental control application that redirects all internet traffic to a remote server for further filtering.
- Make sure users can't intervene in the application's work in any way.

Our team researched different ways this solution could be implemented and came up with several approaches for filtering internet traffic on different Android devices. We also offered an additional solution for filtering traffic in third-party applications.

The result

Apriorit mobile developers created a solution that successfully filters traffic and selects the best internet protection algorithm depending on the device. We implemented all of this functionality as a software development kit (SDK) and thoroughly documented it. This way, our client's internal team was able to successfully integrate the developed functionality into the final product when working on its user interface (UI) and business logic.

Additionally, Apriorit developers created a custom Market service — an alternative to Google Play — containing third-party applications adjusted for traffic filtering. By doing so, it became possible to manage the way children use not only mobile browsers but also some other applications that can connect to the internet.

> *While the product isn't launched yet, the team at Apriorit delivered a bug-free system that will benefit company growth. Their project management style complemented their skilled expertise. While work speed could improve, their dedication to their clients' vision is top-notch.* *Read more »*
>
> **Head of Project Development, Parental Control SaaS Provider**
>
> (Extract from the independent review on Clutch.co)

Finalizing project requirements

After discussing the client's vision of the end product and possible challenges of implementing the desired features, we outlined the following requirements for the application:

Run as an unstoppable VPN service. The application should act as a VPN service for the end user and work without any interruptions once installed and adjusted. It should be impossible for a user to stop or uninstall the application or to change its settings without authorization. The application also should be protected from any traffic interception and it should be impossible to send requests bypassing the application's servers.

Determine the current user. As the same device can be used by both parents and children, the application should be able to detect the current user of the device. Different rules should be applied to different users.

Filter all internet traffic. The application should be able to filter inappropriate web content by intercepting all incoming and outgoing traffic on the Android device. Traffic should be sent to external servers for analysis.

Once traffic is analyzed, the application should provide one of three responses:

- **Allow access** to clean content
- **Allow partial access** to suspicious content
- **Block access completely** to illicit content

Partial access includes modifications to parts of the content considered suspicious. For example, if a naked body is detected, it can be blurred or covered up.

Illicit or restricted content is determined in accordance with the rules defined by the parent. Common examples of such content include pornography, sites of extremist organizations, social networks, and online games.

No content should be accessible without it being checked and filtered on the application's servers.

Forbid unwanted system activity. Any system activities that could lead to bypassing or stopping the application should be forbidden. These activities include but are not limited to:

- entering Safe Mode
- performing a factory reset
- changing device VPN settings
- accessing the Google Play Store.

Block applications. Applications that might provide unauthorized access to the internet can be blocked by the authorized user (the parent).

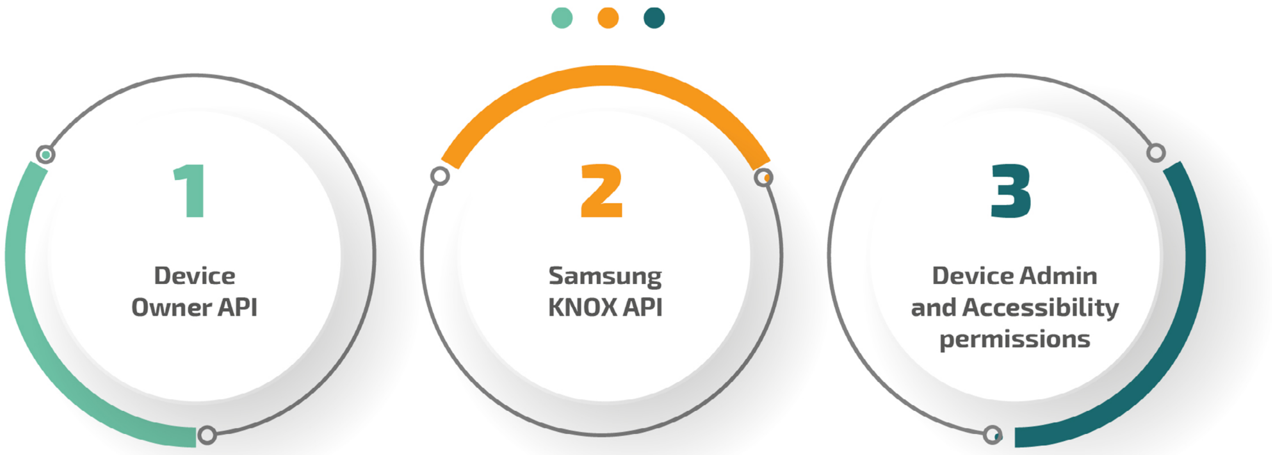
Be implemented as an SDK. Our client wanted their internal team to implement the user interface (UI) and the business logic of the Android application. Therefore, all of the product's functionality needed to be implemented as an SDK and properly documented.

We also needed to research solutions for how to make a parental control app fully protected from unauthorized changes.

Researching approaches to app protection

One of the key goals set for our development team was to make sure that no user would be able to change, disable, or uninstall the application without proper authorization. After preliminary research, we proposed three approaches:

THREE WAYS TO PROTECT THE PARENTAL CONTROL APPLICATION



www.apriorit.com

Using the Device Owner API. Using this Android API, we can make our application a device owner and assign it the rights needed for:

- managing the VPN Service automatically
- storing certificates
- prohibiting the deletion of the application.

The main drawback of this scenario is that before using the Device Owner API, you need to delete all existing accounts on the device. Such a solution isn't something many end users will tolerate.

Using Samsung KNOX. This API provides almost the same capabilities as Device Owner. At the same time, Samsung KNOX doesn't require removing all user accounts from the device, which is a plus.

However, this solution has two major drawbacks:

- It requires you to purchase a license
- It can only be used on Samsung devices

Using Device Admin and Accessibility permissions. It's possible to protect the application using only Device Admin and Accessibility permissions. These device management permissions are easy to obtain, but the implemented solution would be specific to both the vendor and operation system version. To overcome this limitation, additional APIs are required.

As each of these scenarios has certain limitations, we suggested it would be best to implement all three and apply them depending on their availability on a particular device.

Developing the parental control solution

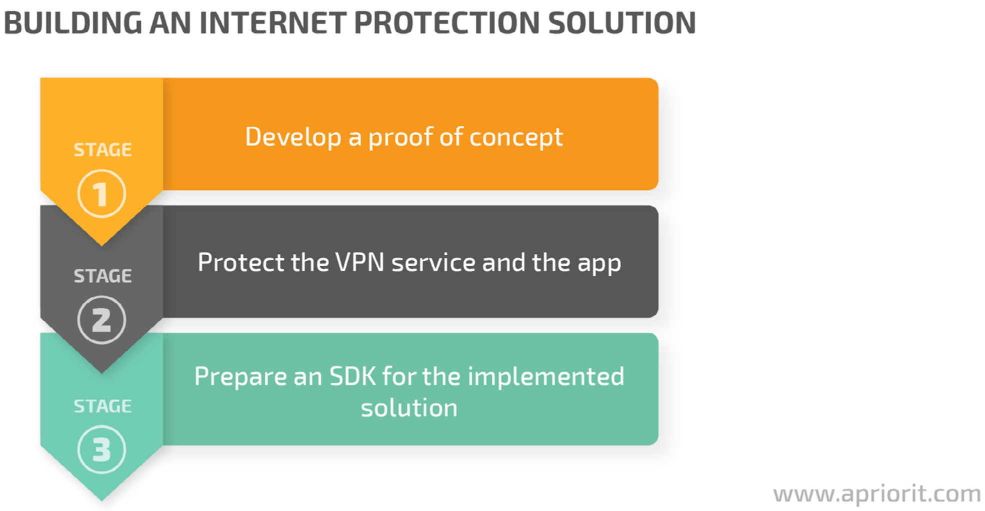
Once we had a clear view of our client's expectations, we started working on the technical implementation of the project. In particular, our dedicated team selected key technologies for the project:

- **Kotlin** as the main parental control app development language
- **Dagger** as the framework for managing dependencies between project entities
- **Room DAO** as the interface for working with databases
- **Ably.io** as the framework for application communications

The overall process of creating this internet protection application was split into two major tasks:

1. Prepare an SDK with the functional part of the parental control application
2. Build a mobile device management (MDM) solution for controlling third-party applications

The first task was split into three stages:


BUILDING AN INTERNET PROTECTION SOLUTION
STAGE 1
Develop a proof of concept
STAGE 2
Protect the VPN service and the app
STAGE 3
Prepare an SDK for the implemented solution
www.apriorit.com

During these stages, the team worked on a traffic filtering Android application that would act as a VPN service and filter traffic in standalone mobile browsers.

The second task was focused on building an MDM solution for managing the traffic of third-party applications with internet access and became the fourth stage of this project.

After each stage, the client evaluated the results and defined the scope for the next stage.

Stage 1. Developing a proof of concept

First, we implemented a proof of concept (PoC) by building a VPN service that intercepts all traffic and redirects it to the client's servers for analysis. Creating a PoC allowed us to:

1. ensure a clear vision of the end result for all parties involved
2. implement Android traffic filtering and analysis functionality that was later used in the final product.

To intercept network traffic, we used Android's base VPNService class coupled with the Redsocks and Netguard open source projects.
We tested the resulting application with different mobile browsers and the traffic was successfully filtered. For example, illicit images found on web pages were properly hidden.

This approach worked well for standalone mobile browsers but wasn't always applicable for third-party applications using certificate pinning. To circumvent this restriction, we proposed building an MDM solution that uses APK patching and implementing a custom Market service.

The Market service, similar to Android's Google Play Store, was needed to give app users access to the patched applications. Our client decided that the team would start working on this MDM solution right after finishing the implementation of the initial parental control app.

Stage 2. Protecting the VPN service and app

No internet protection and parental control application is complete until it's secured from any intentional and unintentional intervention. While parents want to protect their children by setting ground rules and forbidding access to illicit and inappropriate content, children tend to look for ways to evade these constraints. Therefore, the application needed to be protected well enough so that even the most technologically advanced children wouldn't be able to reconfigure, disable, or uninstall it.

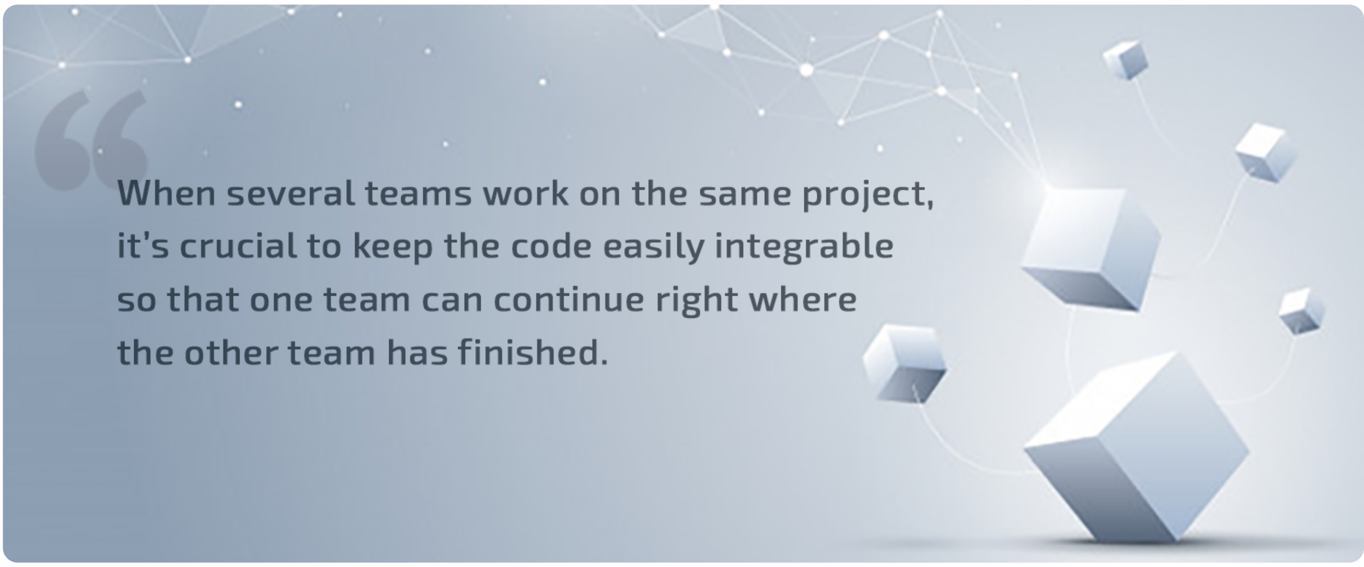
To meet these requirements, we implemented all three previously researched approaches for ensuring the unstoppable work of our VPN service and the app:

- Using the Device Owner API
- Using the Samsung KNOX API
- Using Device Admin and Accessibility permissions

The choice of a particular scenario was device-specific so that we could ensure a smooth user experience and the best protection.

Stage 3. Preparing an SDK for the implemented solution

One of our client's key requirements was to prepare an SDK containing all of the developed functionality. However, in order for an SDK to be usable, it has to be thoroughly documented.



> When several teams work on the same project, it's crucial to keep the code easily integrable so that one team can continue right where the other team has finished.

As our client wanted to entrust the implementation of the UI and business logic to their in-house team, we prepared an SDK with all of the functionality implemented by our specialists during the first two stages. To make sure other developers will experience no complications when using this SDK, we created a library with all our code and documentation describing all interfaces and classes.

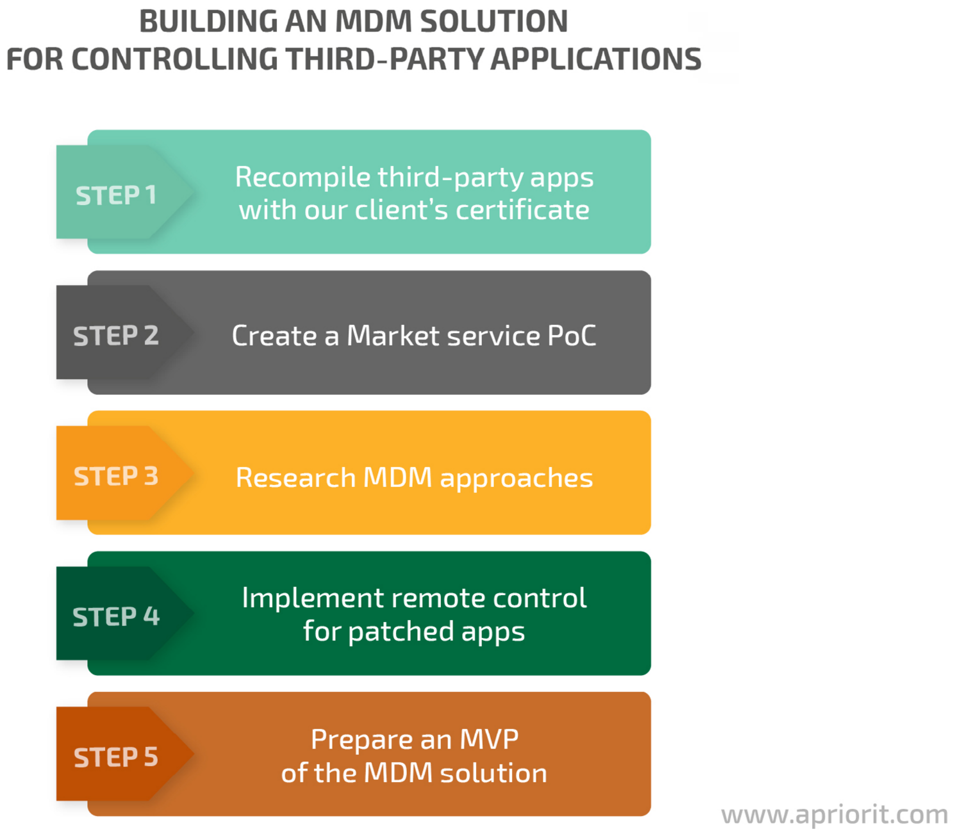
The developers on the client's side successfully used this library when developing the UI.

Stage 4. Preparing an MDM solution for controlling third-party applications

The final part of our research and implementation process was building an MDM solution able to control third-party applications. As we mentioned, we found a VPNService-based solution for filtering internet traffic in browsers. Next, we needed to figure out a way to filter traffic in third-party applications installed on a device.

Standalone mobile browsers are not the only way a user can access the internet. Many social media, instant messenger, and even mobile game applications have embedded browsers that open links directly inside the application. Thus, traffic protection is not complete unless the traffic of all installed third-party applications is also filtered. Unfortunately, due to certificate pinning, this is not as easy as filtering browser traffic.

Apriorit developers implemented this stage of the project in five steps:



BUILDING AN MDM SOLUTION
FOR CONTROLLING THIRD-PARTY APPLICATIONS

STEP 1 Recompile third-party apps with our client's certificate

STEP 2 Create a Market service PoC

STEP 3 Research MDM approaches

STEP 4 Implement remote control for patched apps

STEP 5 Prepare an MVP of the MDM solution

www.apriorit.com

1. Recompile third-party apps with our client's certificate. We researched the possibilities of patching APK files so that applications could access the internet only via the client's application servers. We successfully recompiled more than 50 applications with our client's certificate.

2. Create a Market service PoC. We prepared a PoC for the Market service to be used instead of the restricted Google Play Store. To build this app, we used the F-Droid repository, making some minor modifications such as adjusting scripts and changing icons.

This feature was able to meet two business requirements at once:

- Manage patched and protected applications
- Disable the installation of any unwanted apps from Google Play

3. Research available MDM approaches. We researched the range of basic MDM possibilities without having root access to the device. The scope of our research included such actions as installing and uninstalling applications and blocking application access.

The available solutions resemble those researched at the second stage of the project for protecting the VPN service: the Device Owner API, Samsung KNOX, and Device Admin and Accessibility permissions.

4. Implement remote control for patched apps. Using Ably.io, we implemented a feature to remotely control patched applications installed on a device. We also adjusted the application server so it could send commands to the patched applications.

The list of remote actions included disabling the internet connection on the device, blocking a particular application, and blocking traffic to and from a particular application. This feature also provided parents with an additional control mechanism, allowing them to filter traffic of third-party applications and limit their use on children's devices.

5. Preparing an MVP of the MDM solution. Finally, we prepared a minimum viable product (MVP) with all the functionality implemented during this stage. This solution replaces the Google Play Store on a monitored device with a custom Market service. This service gives children access to over 50 of the most popular applications, which can be managed remotely by their parents.

Challenges and solutions

When developing a parental control app for Android, our dedicated Apriorit team faced a number of challenges:

- **Difficulties with app installation and configuration.** Our client wanted the final application to be as easy to install and configure as possible so a user doesn't need to interact with it too much. Enabling the app to install certificates could help reduce the number of such interactions. However, the developed application didn't have enough rights to install application certificates.

On Samsung devices, we tackled this challenge with the help of Samsung KNOX. On the majority of other devices, we used the Accessibility service to overcome this limitation.

- **Application support challenges.** Depending on the device, the Accessibility service works differently. This adds complexity to application support, as we need to adjust to the peculiarities of every specific device. As there's no actual solution for this problem, the need for extra efforts during support should be taken into account.

- **Traffic monitoring limitations.** The feature responsible for pinning certificates doesn't allow for monitoring the traffic of some applications. Our client decided to continue using this feature as is.

The impact

During the first three stages of the project, Apriorit developers created an internet protection app that our client was completely satisfied with. They plan to place the created parental control application on Google Play and sell it to end users.

The fourth stage of developing the MDM solutions is still in progress, as we need to improve the competitiveness of the custom Market service. To do this, we need to patch more applications with different levels of popularity and add them to the Market.