

Perform 2018

Mobile Performance Management for Native Mobile Applications

Hands-on Training Instructions

Lesson 2 – Auto-instrumentation in Android Studio





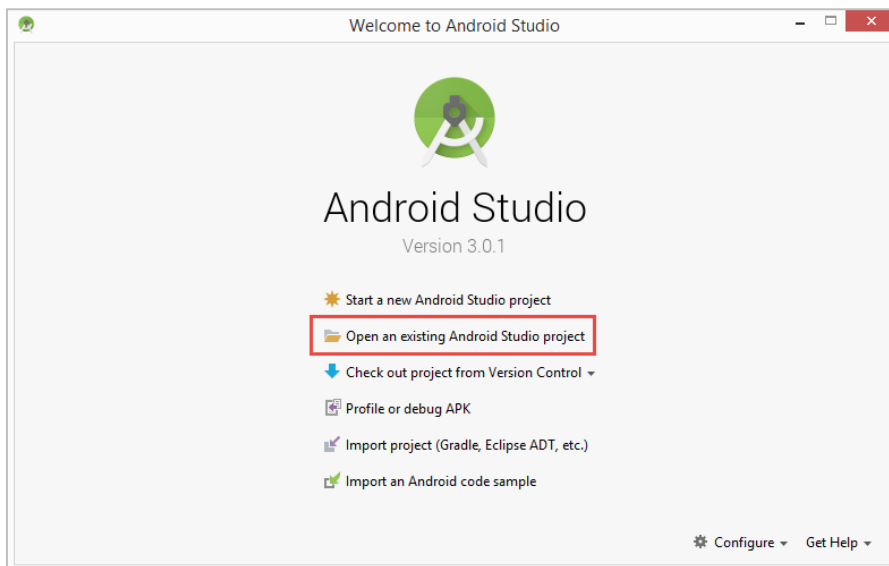
Preparation - Download the mobile applications projects

1. Launch your command line interface
 - a. Windows : Windows button -> type "cmd" at the prompt
 - b. MacOS : Search for "Terminal" in the Spotlight Search
2. Create a directory structure for your projects. **It is important that the directory path does not contain any space.**
For example, create directly from the C drive or from your MacOS drive
 - a. Windows : **md c:\projects**
 - b. MacOS : **mkdir projects**
3. Switch to your projects directory :
 - a. Windows : **cd c:\projects**
 - b. MacOS : **cd projects**
4. Clone the mobile-hotday-2018 repository from Github
 - a. git clone <https://github.com/Dynatrace/mobile-hotday-2018.git>

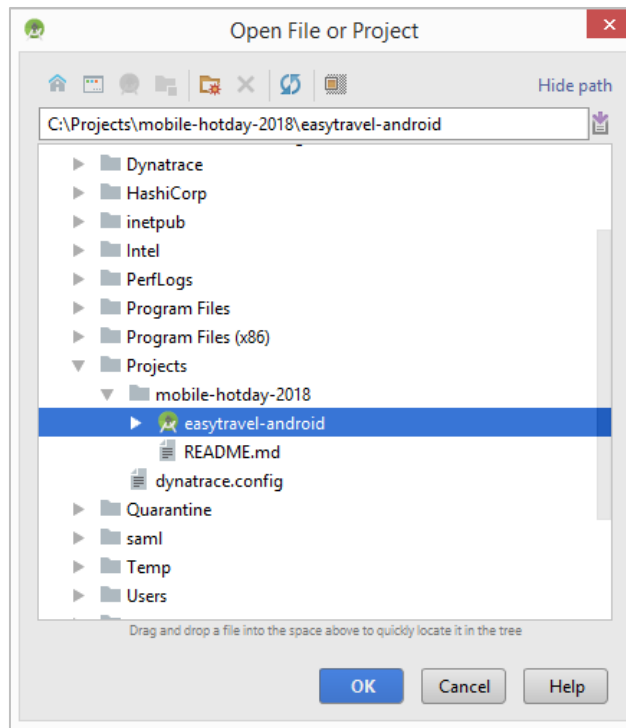
Lesson 2 - Auto-instrumentation in Android Studio

Importing the project in Android Studio

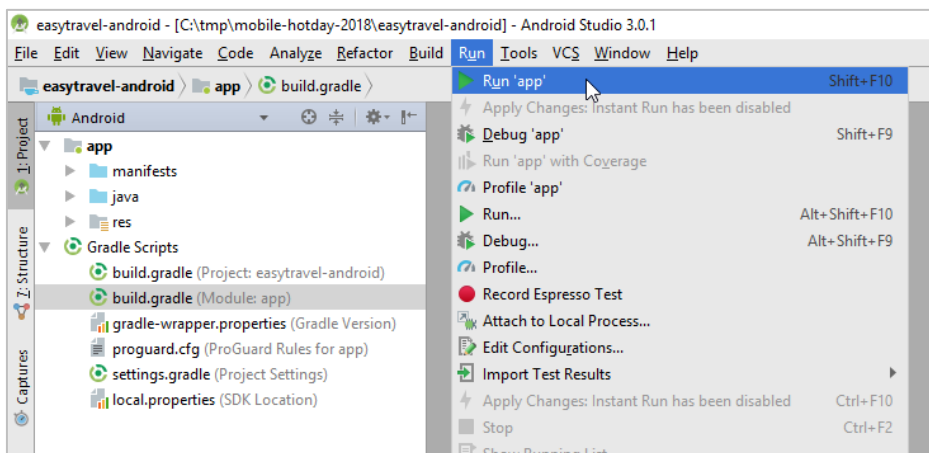
1. Launch Android Studio
2. Select **Open an existing Android Studio project**



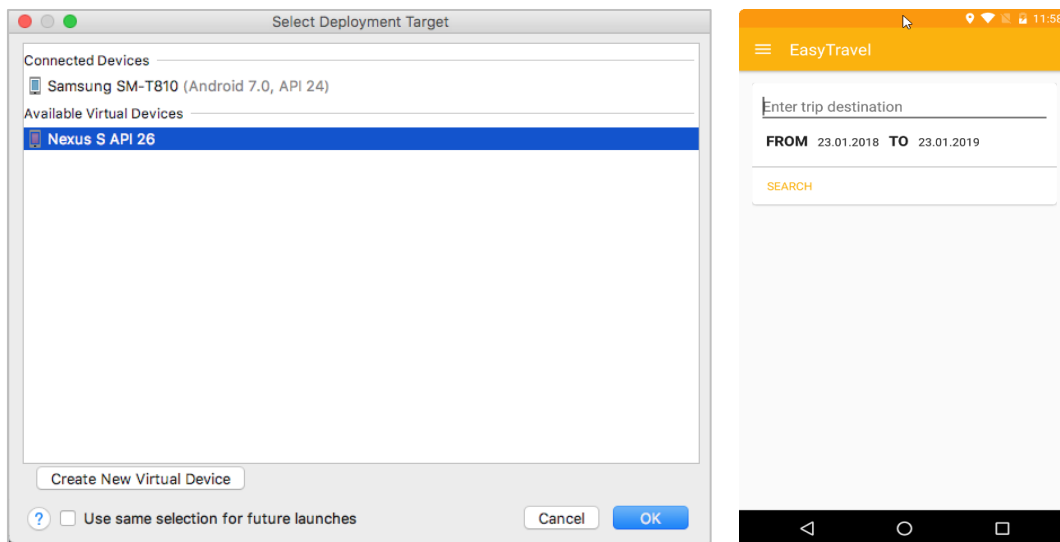
3. Browse to your project directory and select the **easytravel-android** directory



4. Wait for Gradle to complete the build (look for “Gradle build finished” at the bottom left, in the status bar). Run the **easyTravel** application by clicking on the **Run** button.
- 5.

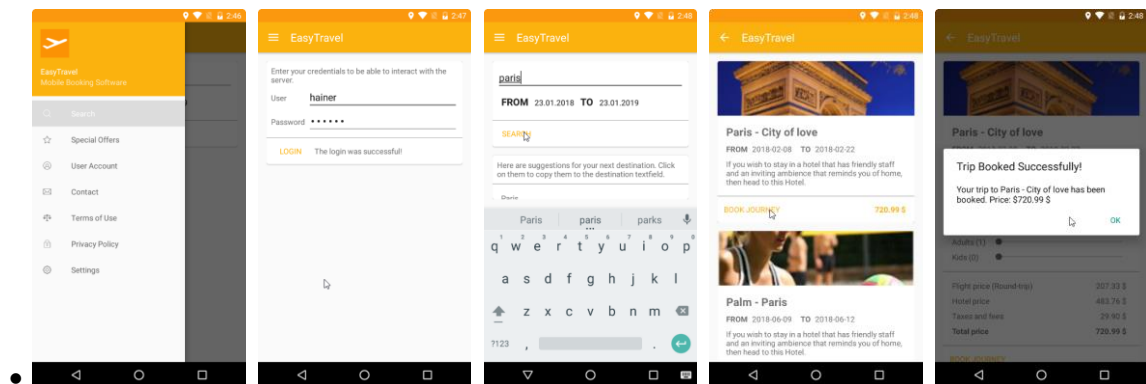


6. Select a device (either physical or virtual) to deploy the application.



7. Navigate and explore the functionalities of the application

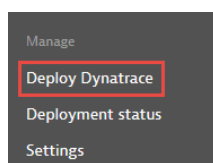
- Log in
- Book a journey (e.g. search for Paris)
- View the special offers
- Settings will be used later in the exercise to simulate problems



8. Although it is optional, you might want to uninstall the application from the device before we instrument it.

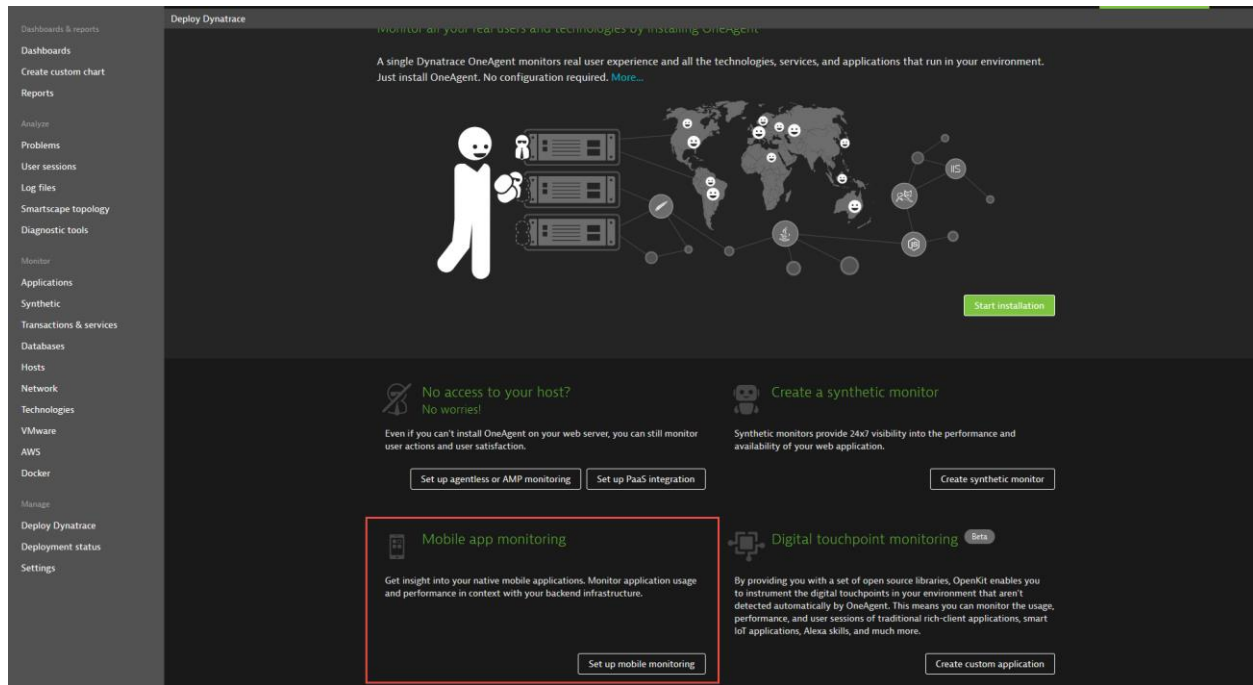
Instrumenting the application with Dynatrace

9. Now, you will add the configuration to have Dynatrace monitor the application. First, we will define the native mobile application. In the Dynatrace portal, on the left menu at the bottom, click on **Deploy Dynatrace**





10. On the Deploy Dynatrace screen, go to Mobile app monitoring and click on **Set up mobile monitoring**





11. Enter an application name. For consistency, please use the following syntax: YourName_Native (replace the YourName string by your First_LastName 😊). Click on **Create mobile app**.
12. Click on the Google Android platform button. The screen will display the snippet that needs to be copied in your Android project **build.gradle** script



Instrument your application to prepare for monitoring.

Select your platform to begin:


Google Android


Apple iOS

Select method for instrumentation:

Gradle

Command line

Modify your **build.gradle** script

Update your module's **build.gradle** script to include the **jcenter()** repository and apply the Dynatrace plugin (requires Android Plugin for Gradle version 1.5 or higher). The generated **applicationId** is unique for this mobile application.

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.dynatrace.tools:android:+'
    }
}

apply plugin: 'com.dynatrace.tools:android'
dynatrace {
    defaultConfig {
        applicationId '3fdeefbb-6cfe-45f4-9085-e50f3fb668cc'
        environmentId 'bf32559dsi'
        cluster 'https://bf-sprint.dynatracelabs.com'
    }
}
```

Copy

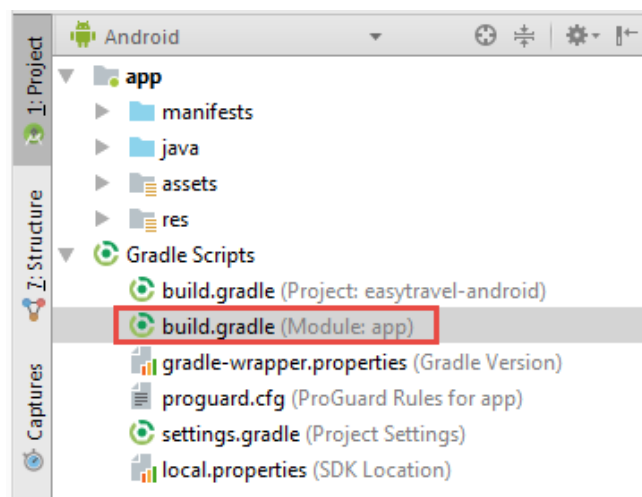
You can now build and run your app using your IDE.

Android Studio 2.0 users:

Please disable the **Instant Run** setting. To do this, open the **Settings** and navigate to **Build, Execution, Deployment > Instant Run**. Then deselect the **Enable Instant Run** checkbox.

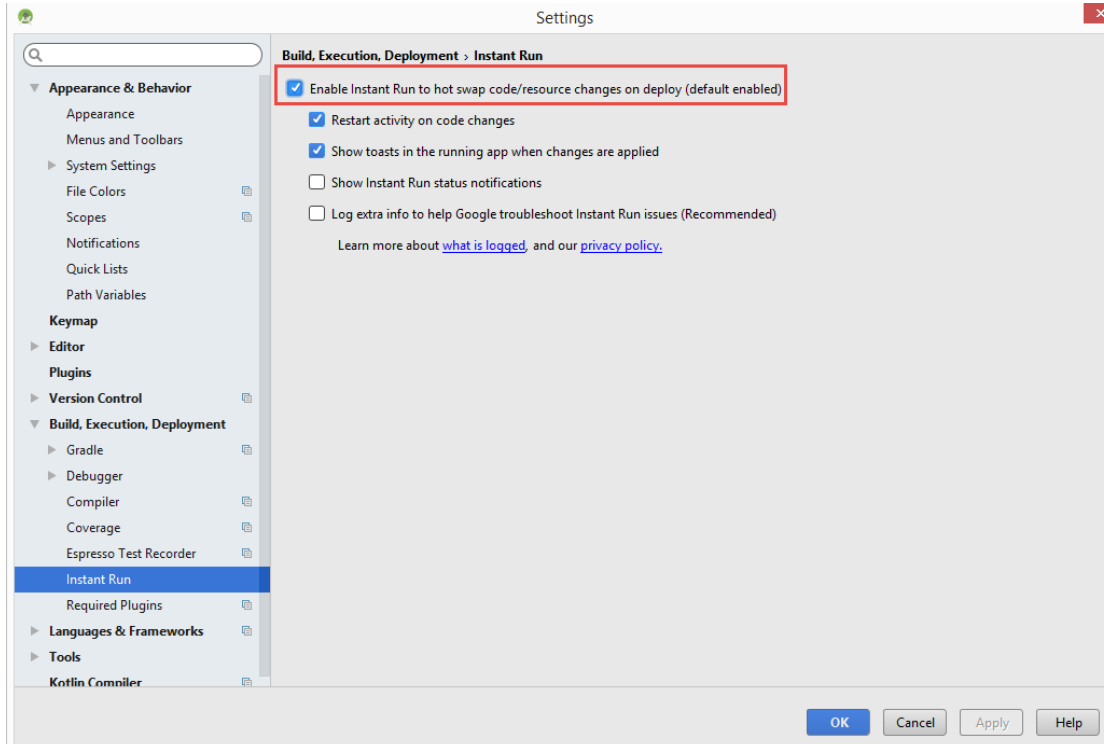
13. In Android Studio, open the **build.gradle** script.

1. There will be 2 **build.gradle** scripts. Use the one for the module.

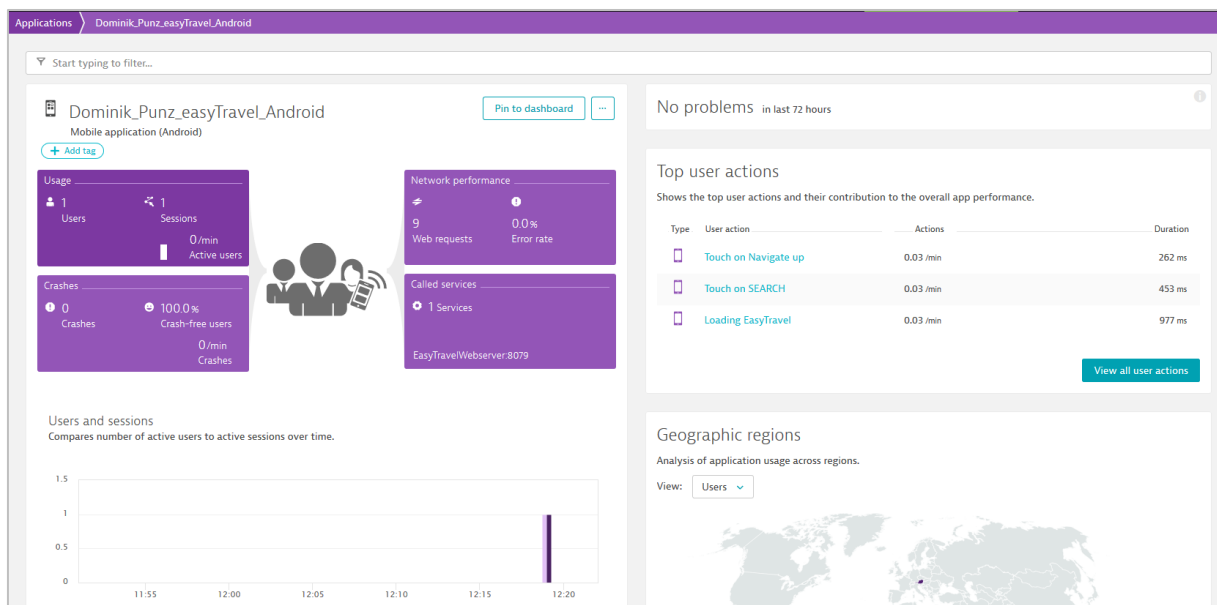


14. Find **TODO (1)** in the gradle script and insert the snippet you copied from Dynatrace.

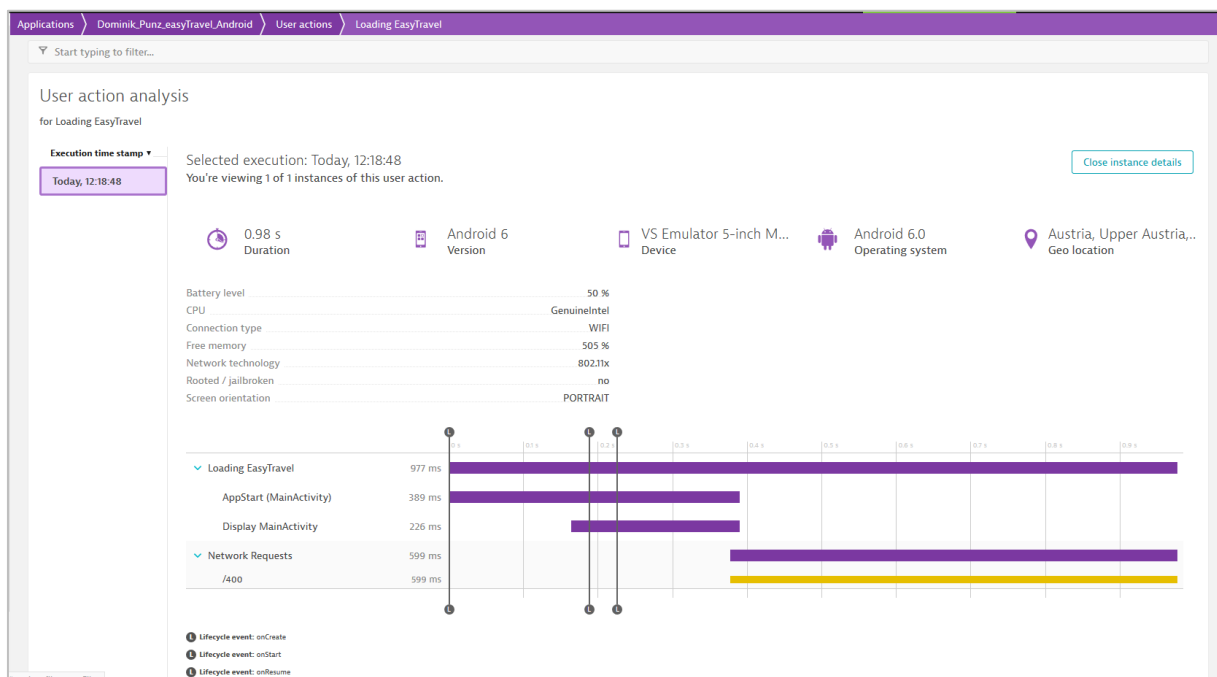
15. **Warning:** the Android agent is not compatible with the Android Studio **Instant Run** feature. Before running the application on a device, you need to make sure **Instant Run** is disabled. In the Android Studio menu go to **File->Settings**. In the settings vertical menu, go to **Build, Execution, Deployment->Instant Run**. Uncheck the **Enable Instant Run...** check box. Click OK.



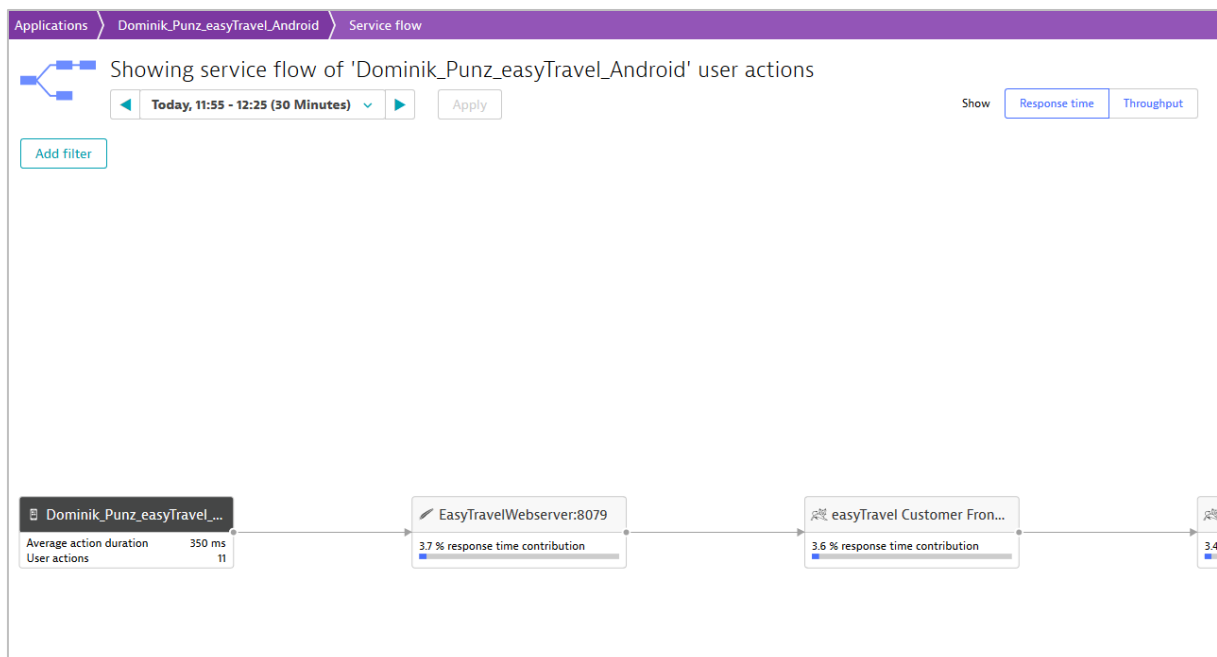
16. Wait for the Gradle sync to complete and launch the application by clicking on the run button and select a device to deploy it.
17. In the Gradle console you will notice output from the auto-instrumentor.
 - Search for Paris and try to book the journey.
 - Then close the app.
 - This will trigger the mobile agent to send the data to Dynatrace.
18. Return to the Dynatrace console, close the settings and go to your application screen by clicking the bread crumb and wait for about 2 minutes. You will then see the first details flowing into the app screen.
19. View the usage statistics, network performance, crashes and services section.



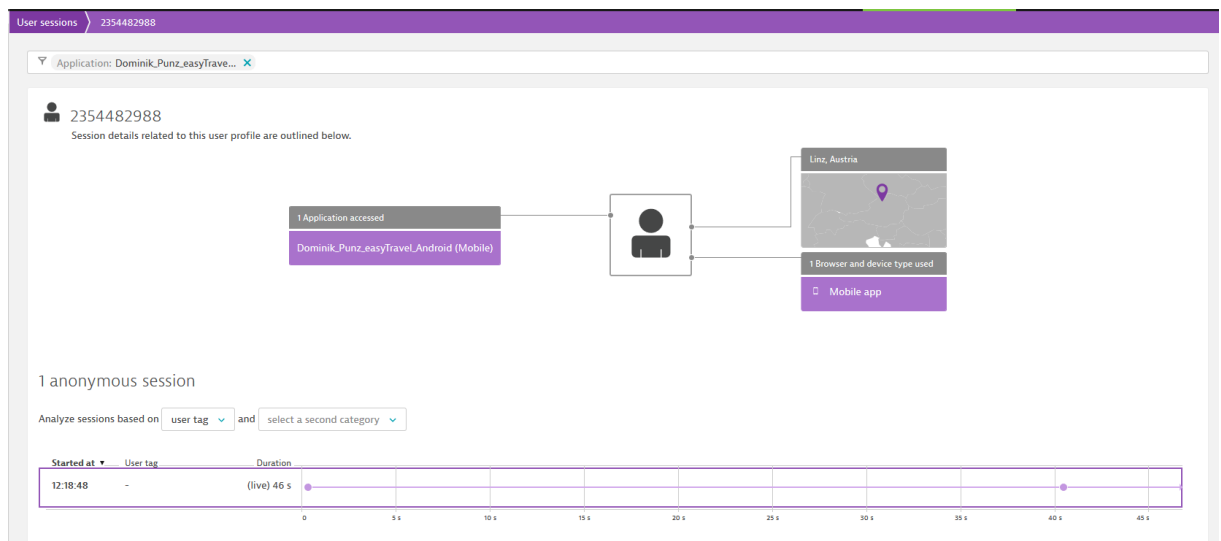
20. Analyze the user actions and locate the **Loading EasyTravel** user action and view the details.



21. Go back to your application screen (again, you can use the breadcrumb), click on **Called Services** in the infographic and view the **Service Flow**.

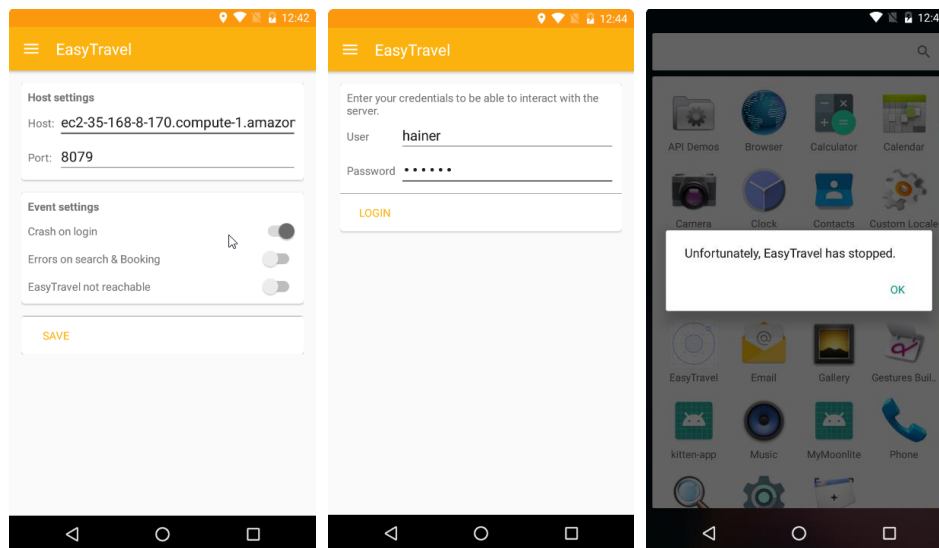


22. Go back to the application main screen and click on **Analyze user sessions** (button at the bottom – you might need to scroll down).

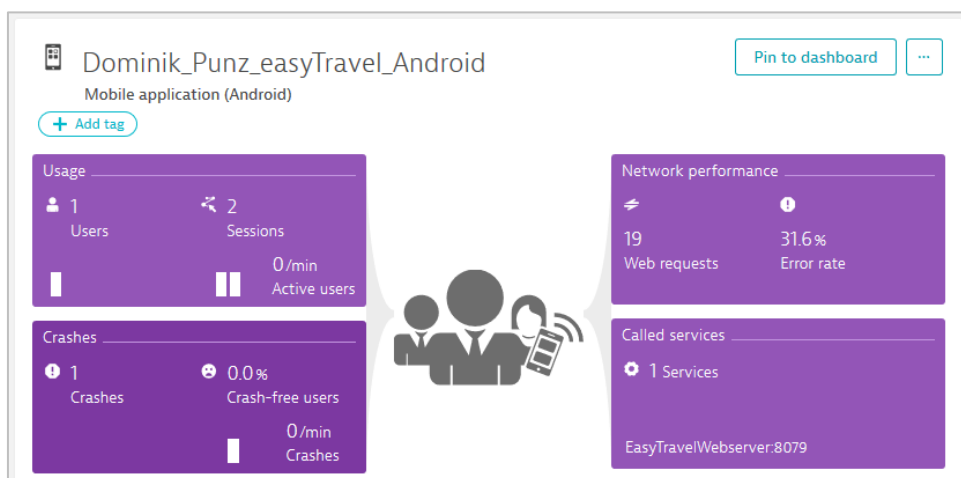


Crash Reporting

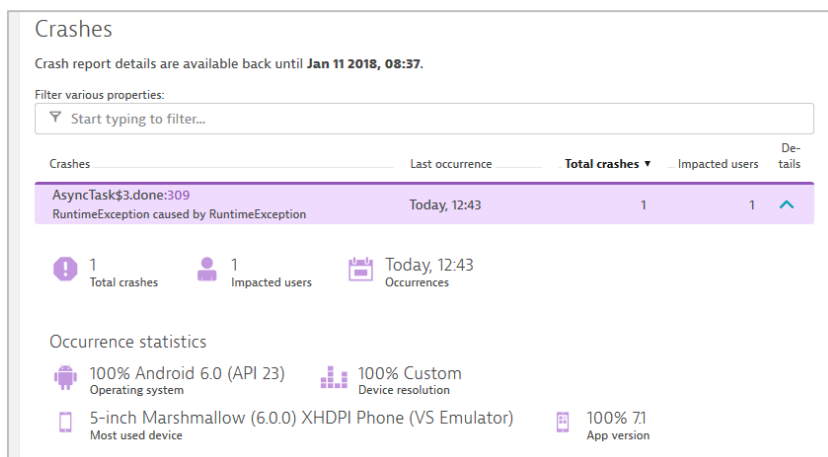
23. Return to Android Studio and run the application again.
24. In the menu, select **Settings** and enable **Crash on login** and click the **Save** button.
25. In the menu, select **Login**, enter a user name and password (for example, maria / maria) and click **Login**.



26. Return to the Dynatrace console and wait 2 minutes. The crash will show up in the infographic.



27. View the crash details and notice that the stack trace is readable.





28. Often, mobile applications are obfuscated so the crash stack trace is not readable. Let's modify the application and use Proguard to obfuscate it. In Android Studio, go to the module gradle script and locate the **TODO (2)**.
29. Enable Proguard in the **debug** build by uncommenting the 2 lines (remove the **//**) under **TODO (2)** and sync Gradle.

```
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

android {
    apply plugin: 'com.android.application'

    android {
        compileSdkVersion 23
        buildToolsVersion '26.0.2'

        defaultConfig {
            applicationId "com.dynatrace.easytravel.android"
            minSdkVersion 17
            targetSdkVersion 23
            javaCompileOptions {
                annotationProcessorOptions {
                }
            }
        }

        buildTypes {
            release {
                minifyEnabled true
                proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard.cfg'
            }
            debug {
                // TODO (2) enable proguard in debug build, run it in the emulator and generate a crash
                minifyEnabled true
                proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard.cfg'
            }
        }
    }
}
```

30. Run the app again, trigger the crash and go back to Dynatrace. After 2 minutes the crash will show up. Notice the difference to the first crash. This time it is obfuscated.

Crashes

Crash report details are available back until Jan 11 2018, 08:37.

Filter various properties:
Start typing to filter...

Crashes	Last occurrence	Total crashes	Impacted users	Details
AsyncTask\$3.done:309 RuntimeException caused by RuntimeException	Today, 12:43	1	1	⌵
UserFragment\$a.a:105 RuntimeException caused by RuntimeException	Today, 12:57	1	1	⬆

1 Total crashes

1 Impacted users

Today, 12:57 Occurrences

Occurrence statistics

100% Android 6.0 (API 23)
Operating system

100% Custom
Device resolution

5-inch Marshmallow (6.0.0) XHDPi Phone (VS Emulator)
Most used device

100% 71
App version

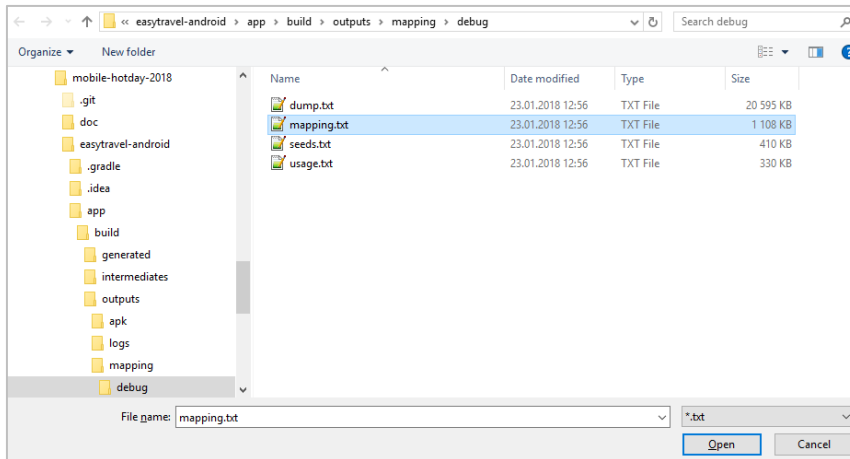
Stacktrace (Version 71; Build 20170129) [Download stacktrace](#) [Upload mapping file](#)

```
java.lang.RuntimeException: An error occurred while executing doInBackground()
    at android.os.AsyncTask$3.done (AsyncTask.java:309)
    at java.util.concurrent.FutureTask.finishCompletion (FutureTask.java:354)
    at java.util.concurrent.FutureTask.setException (FutureTask.java:223)
    at java.util.concurrent.FutureTask.run (FutureTask.java:242)
    at android.os.AsyncTask$SerialExecutor$1.run (AsyncTask.java:234)
    at java.util.concurrent.ThreadPoolExecutor.runWorker (ThreadPoolExecutor.java:1113)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run (ThreadPoolExecutor.java:588)
    at java.lang.Thread.run (Thread.java:818)

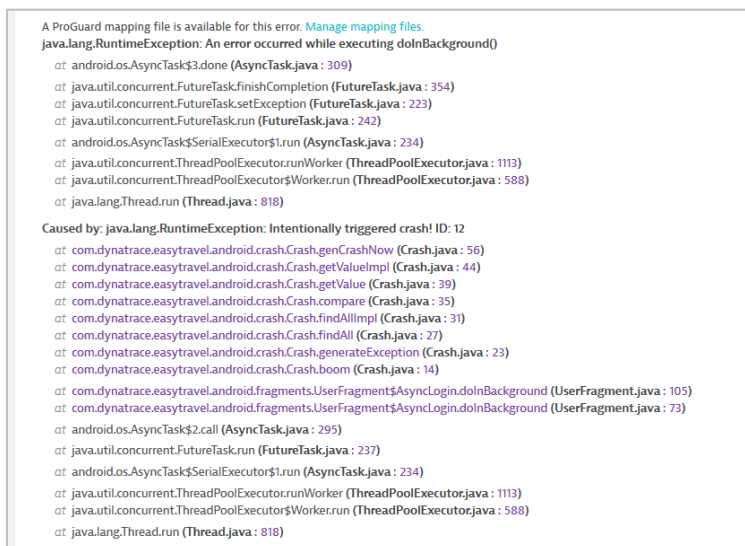
Caused by: java.lang.RuntimeException: Intentionally triggered crash! ID: 12
    at cs.j (SourceFile: 56)
    at cs.h (SourceFile: 44)
    at cs.g (SourceFile: 39)
    at cs.f (SourceFile: 35)
    at cs.e (SourceFile: 31)
    at cs.d (SourceFile: 27)
    at cs.c (SourceFile: 23)
    at cs.a (SourceFile: 14)
    at com.dynatrace.easytravel.android.fragments.UserFragment$a.a (SourceFile: 105)
    at com.dynatrace.easytravel.android.fragments.UserFragment$a.doInBackground (SourceFile: 73)
    at android.os.AsyncTask$2.call (AsyncTask.java:295)
    at java.util.concurrent.FutureTask.run (FutureTask.java:237)
    at android.os.AsyncTask$SerialExecutor$1.run (AsyncTask.java:234)
    at java.util.concurrent.ThreadPoolExecutor.runWorker (ThreadPoolExecutor.java:1113)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run (ThreadPoolExecutor.java:588)
    at java.lang.Thread.run (Thread.java:818)
```



31. Click on **Upload mapping file** and locate the **mapping.txt** file in the **easytravel-android/app/build/outputs/mapping/debug** directory.



32. View the unobfuscated stack trace in Dynatrace



33. View the list of available mapping files

