Perform 2018
Mobile Performance Management for Native Mobile Applications

Hands-on Training Instructions

Lesson 2 – Auto-instrumentation for Android Native Mobile Application

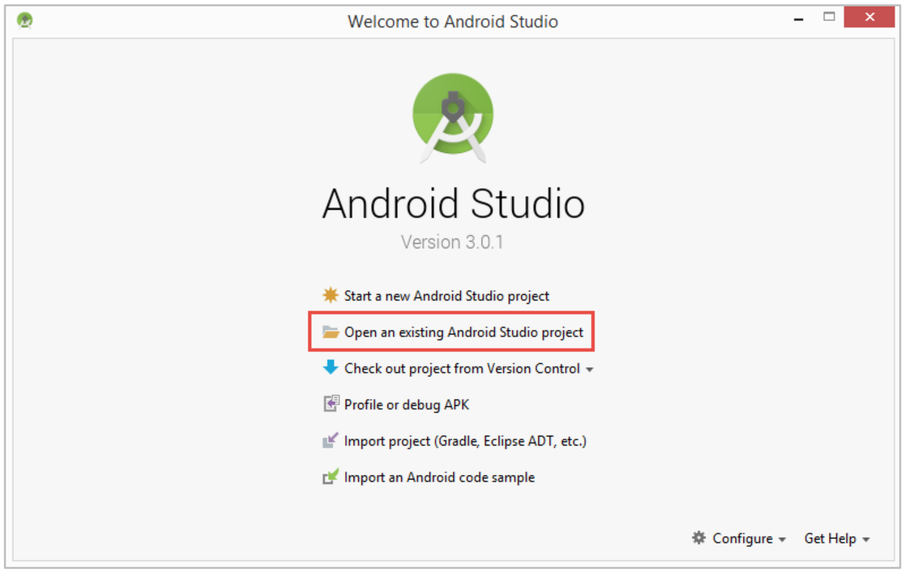
Preparation - Download the mobile applications projects

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

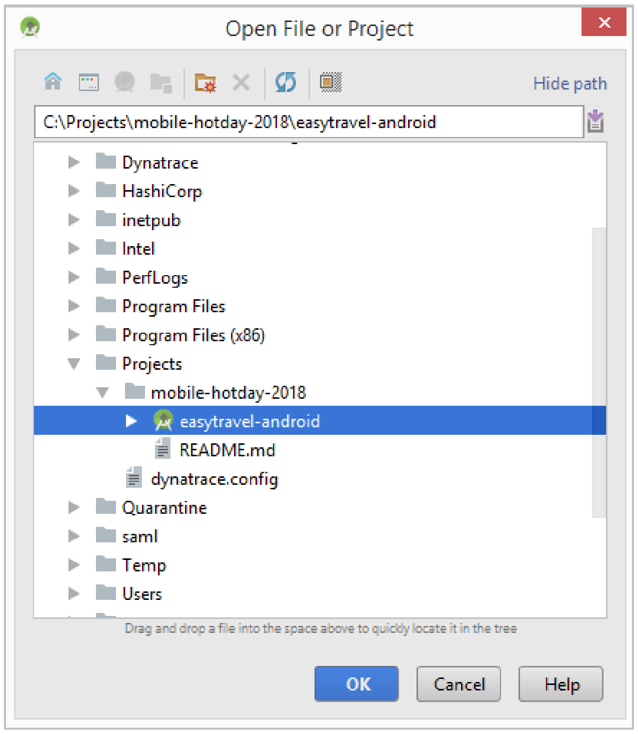
Lesson 2 - Auto-instrumentation for Android Native Mobile Application

Importing the project in Android Studio

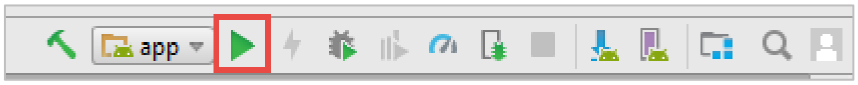
1. Launch Android Studio
2. Select **Open an existing Android Studio project**

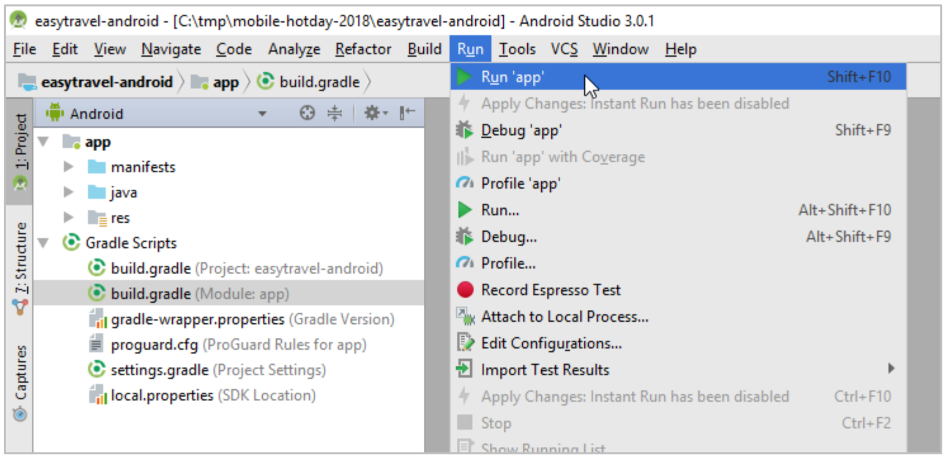

Welcome to Android Studio
Android Studio
Version 3.0.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

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

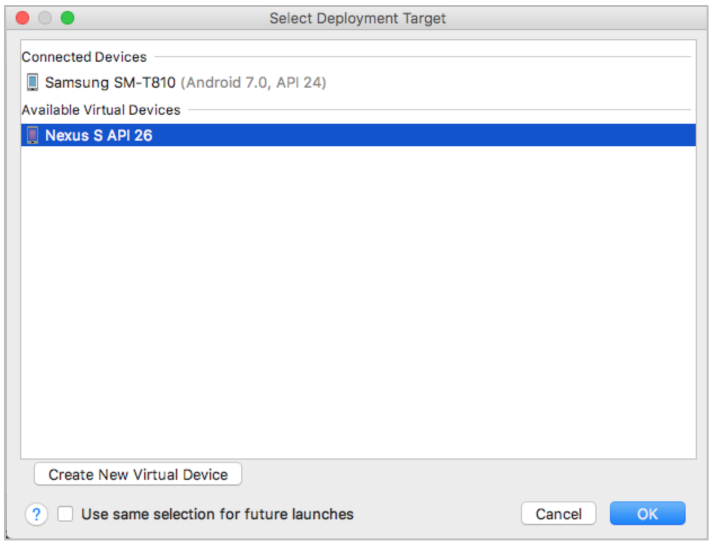


4. Sync Gradle (click on **Sync Now** on top-right of the editor). Run the **easyTravel** application by clicking on the **Run** button.
5.




easytravel-android - [C:\tmp\mobile-hotday-2018\easytravel-android] - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
easytravel-android app build.gradle
Android
app
manifests
java
res
Gradle Scripts
build.gradle (Project: easytravel-android)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard.cfg (ProGuard Rules for app)
settings.gradle (Project Settings)
local.properties (SDK Location)
Run 'app' Shift+F10
Apply Changes: Instant Run has been disabled
Debug 'app' Shift+F9
Run 'app' with Coverage
Profile 'app'
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Profile...
Record Espresso Test
Attach to Local Process...
Edit Configurations...
Import Test Results
Apply Changes: Instant Run has been disabled Ctrl+F10
Stop Ctrl+F2
Show Running List
1: Project
7: Structure
Captures

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


Select Deployment Target

Connected Devices

Samsung SM-T810 (Android 7.0, API 24)

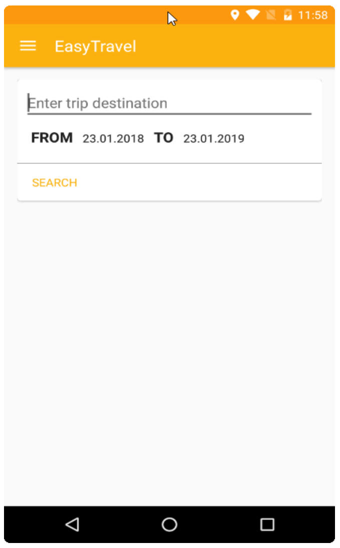
Available Virtual Devices

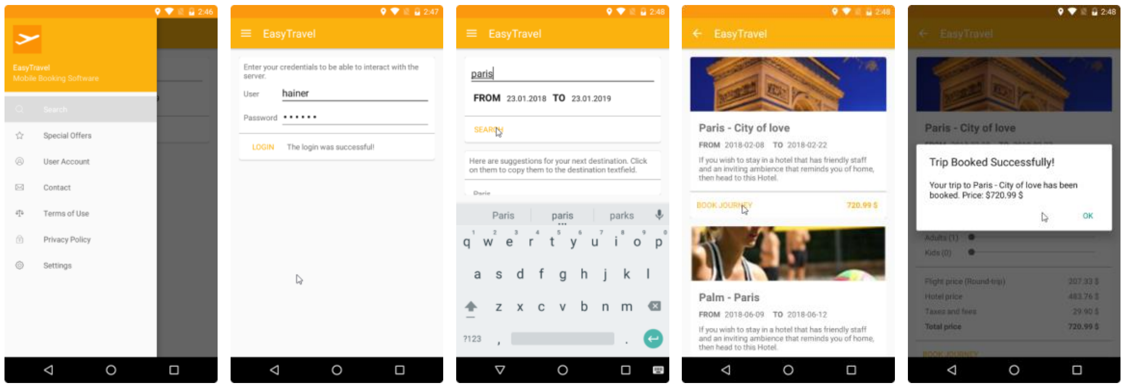
Nexus S API 26

Create New Virtual Device

Use same selection for future launches

Cancel OK

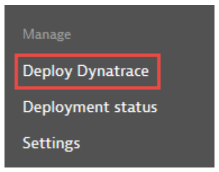


7. Navigate and explore the functionalities of the application
   - Log in
   - Book a journey (e.g. search for Paris)
   - View the special offers
   - Settings will be used later in the exercise to simulate problems
   - 

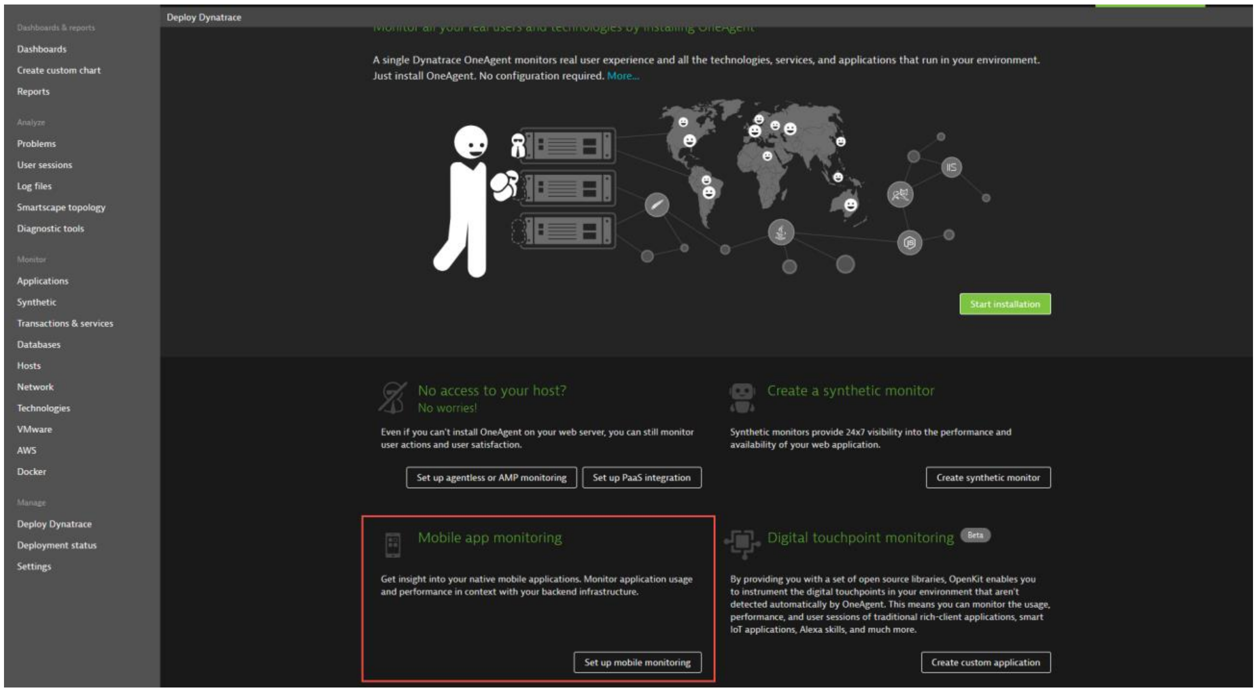
8. Although it is optional, you might want to uninstall the application from the device before we instrument it.

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**



A single Dynatrace OneAgent monitors real user experience and all the technologies, services, and applications that run in your environment. Just install OneAgent. No configuration required. More...

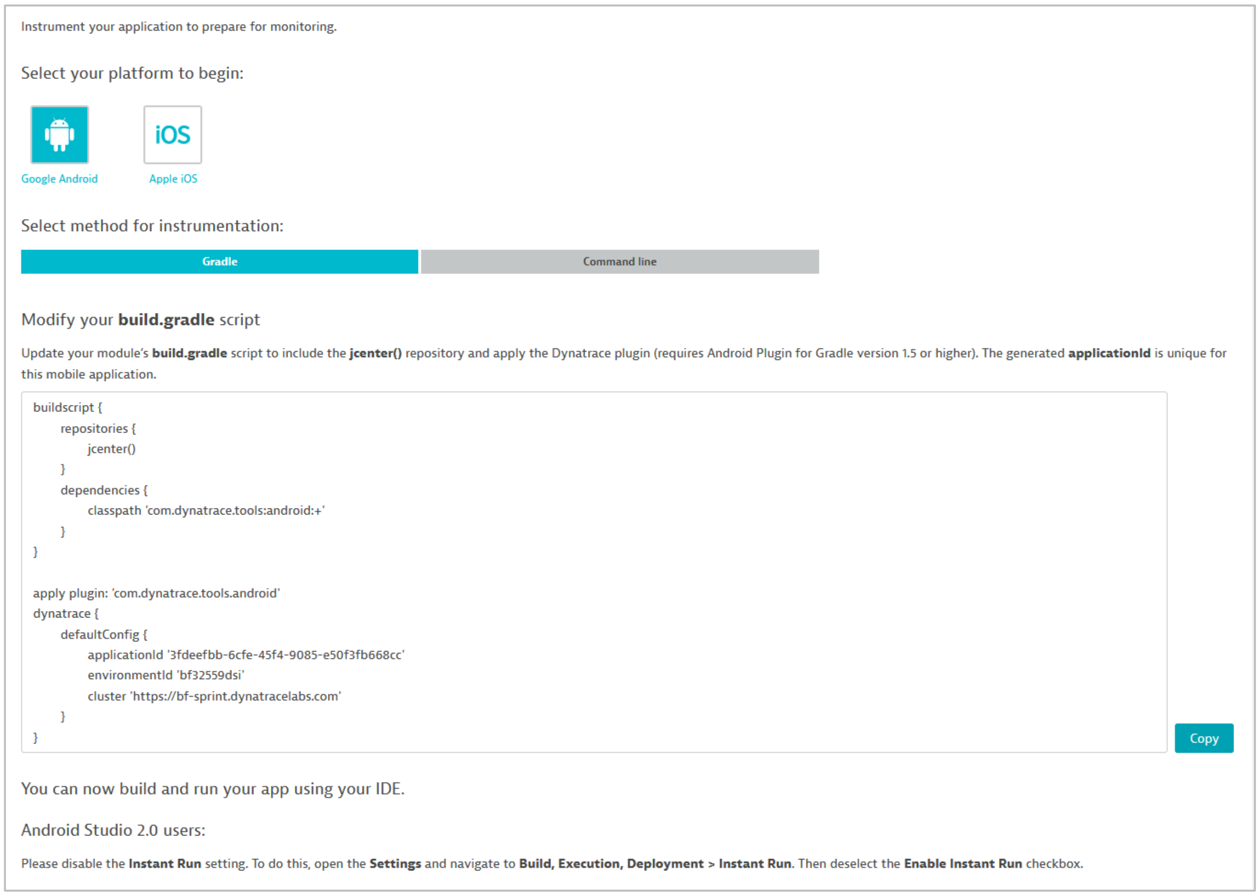
No access to your host?
No worries!
Even if you can't install OneAgent on your web server, you can still monitor user actions and user satisfaction.

Create a synthetic monitor
Synthetic monitors provide 24x7 visibility into the performance and availability of your web application.

Mobile app monitoring
Get insight into your native mobile applications. Monitor application usage and performance in context with your backend infrastructure.

Digital touchpoint monitoring
By providing you with a set of open source libraries, OpenKit enables you to instrument the digital touchpoints in your environment that aren't detected automatically by OneAgent. This means you can monitor the usage, performance, and user sessions of traditional rich-client applications, smart IoT applications, Alexa skills, and much more.

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:

Select method for instrumentation:

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.

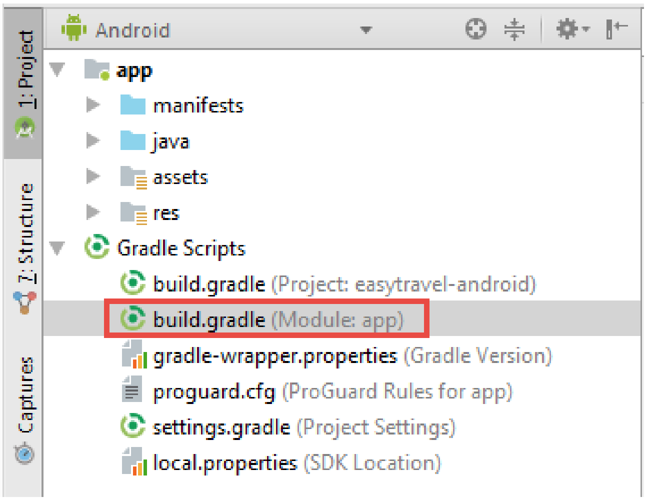
```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.dynatrace.tools:android:+'
    }
}

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

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

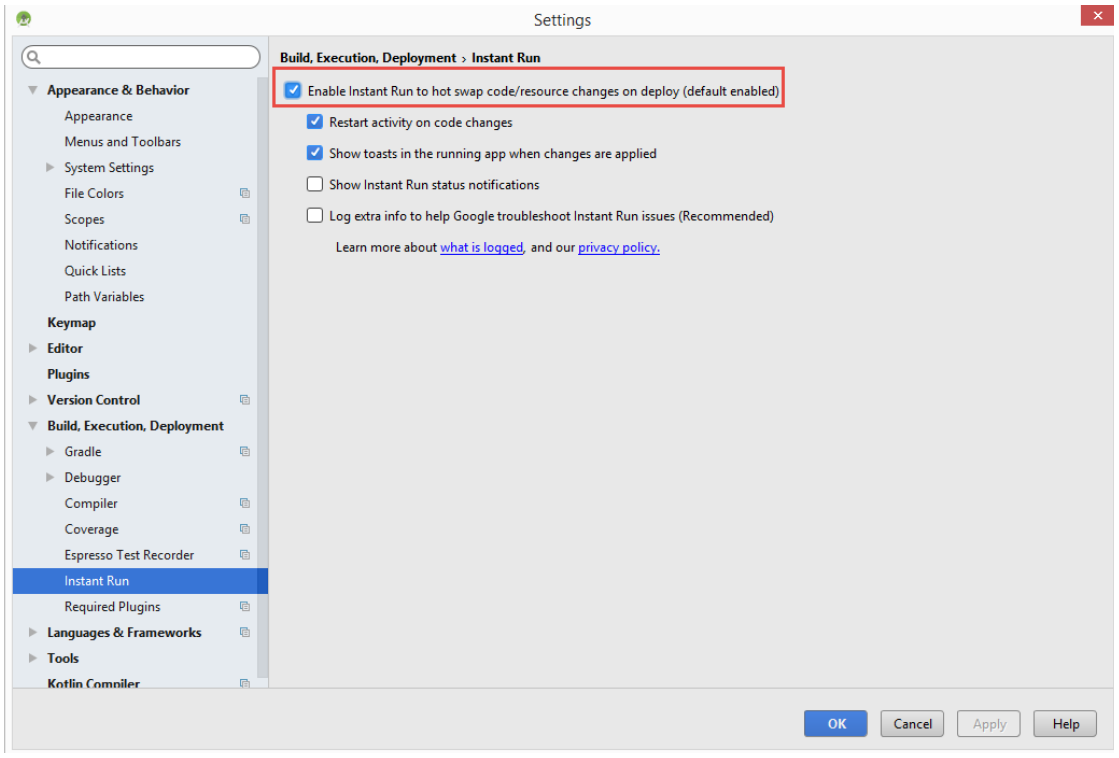
Android Studio 2.0 users:

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.


1: Project
7: Structure
Captures
Android
app
manifests
java
assets
res
Gradle Scripts
build.gradle (Project: easytravel-android)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard.cfg (ProGuard Rules for app)
settings.gradle (Project Settings)
local.properties (SDK Location)

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 **Files->Settings**. In the settings vertical menu, go to **Build, Execution, Deployment->Instant Run**. Uncheck the **Enable Instant Run…** check box. Click OK.


Settings

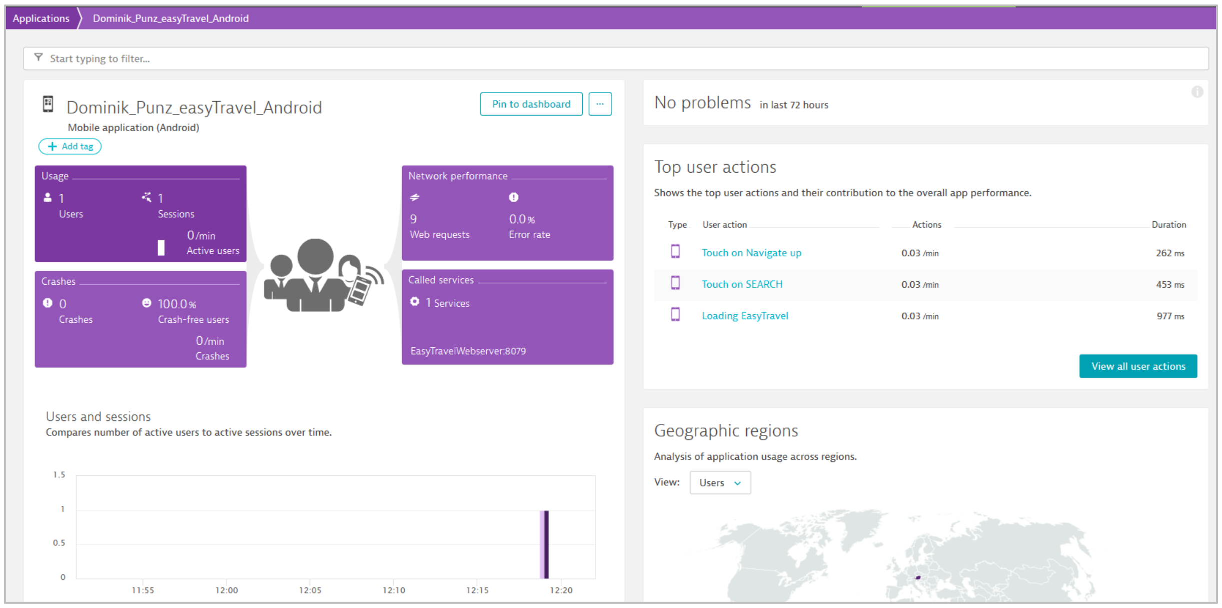
Build, Execution, Deployment › Instant Run

- [✓] Enable Instant Run to hot swap code/resource changes on deploy (default enabled)
  - [✓] Restart activity on code changes
  - [✓] Show toasts in the running app when changes are applied
  - [ ] Show Instant Run status notifications
  - [ ] Log extra info to help Google troubleshoot Instant Run issues (Recommended)

  Learn more about what is logged, and our privacy policy.

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.



Applications 〉 Dominik_Punz_easyTravel_Android

Start typing to filter...

Dominik_Punz_easyTravel_Android
Mobile application (Android)
+ Add tag

Pin to dashboard ···

Usage
1 Users
1 Sessions
0 /min Active users

Crashes
0 Crashes
100.0 % Crash-free users
0 /min Crashes

Network performance
9 Web requests
0.0 % Error rate

Called services
1 Services
EasyTravelWebserver:8079

No problems in last 72 hours

Top user actions
Shows the top user actions and their contribution to the overall app performance.

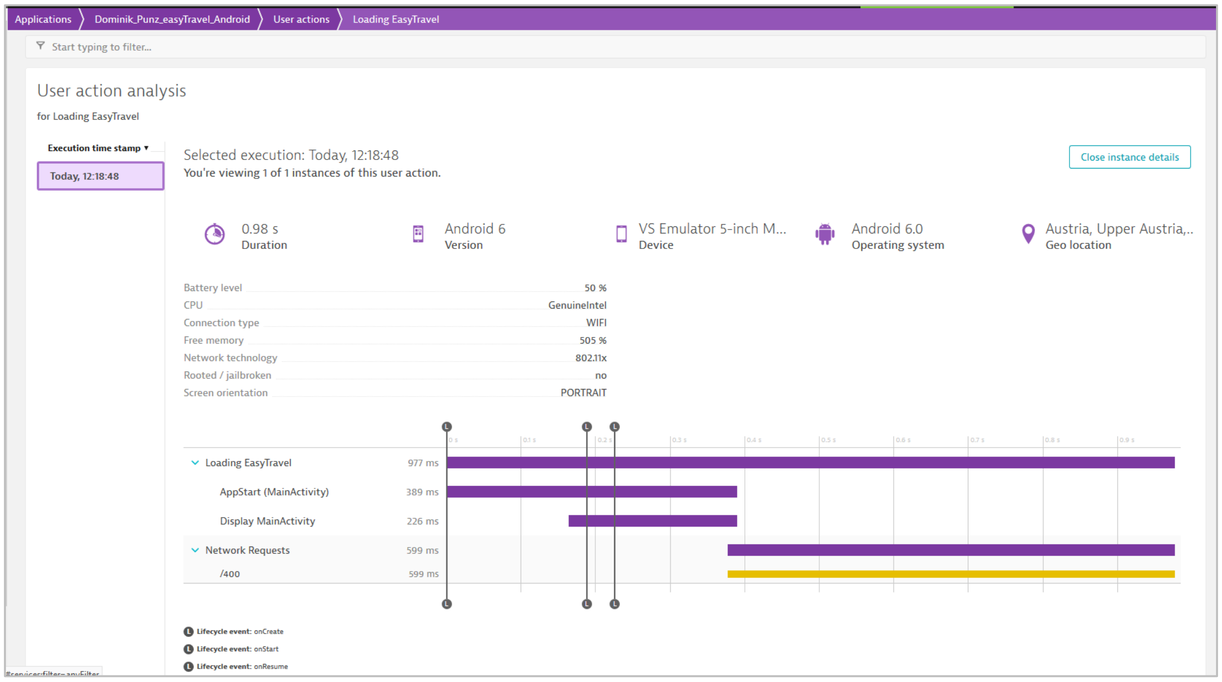
Type	User action	Actions	Duration
▯	Touch on Navigate up	0.03 /min	262 ms
▯	Touch on SEARCH	0.03 /min	453 ms
▯	Loading EasyTravel	0.03 /min	977 ms

View all user actions

Users and sessions
Compares number of active users to active sessions over time.

Geographic regions
Analysis of application usage across regions.
View: Users

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



Applications 〉 Dominik_Punz_easyTravel_Android 〉 User actions 〉 Loading EasyTravel

Start typing to filter...

User action analysis
for Loading EasyTravel

Execution time stamp ▾
Today, 12:18:48

Selected execution: Today, 12:18:48
You're viewing 1 of 1 instances of this user action.

Close instance details

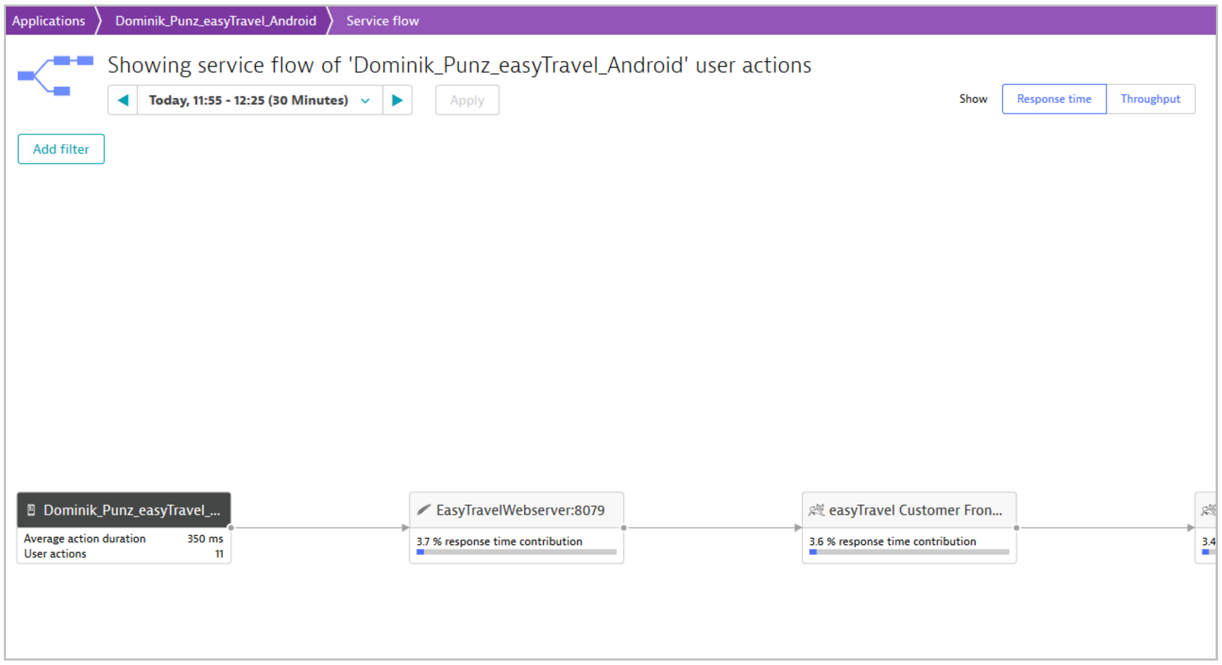
0.98 s Duration
Android 6 Version
VS Emulator 5-inch M... Device
Android 6.0 Operating system
Austria, Upper Austria,.. Geo location

Battery level	50 %
CPU	GenuineIntel
Connection type	WIFI
Free memory	505 %
Network technology	802.11x
Rooted / jailbroken	no
Screen orientation	PORTRAIT

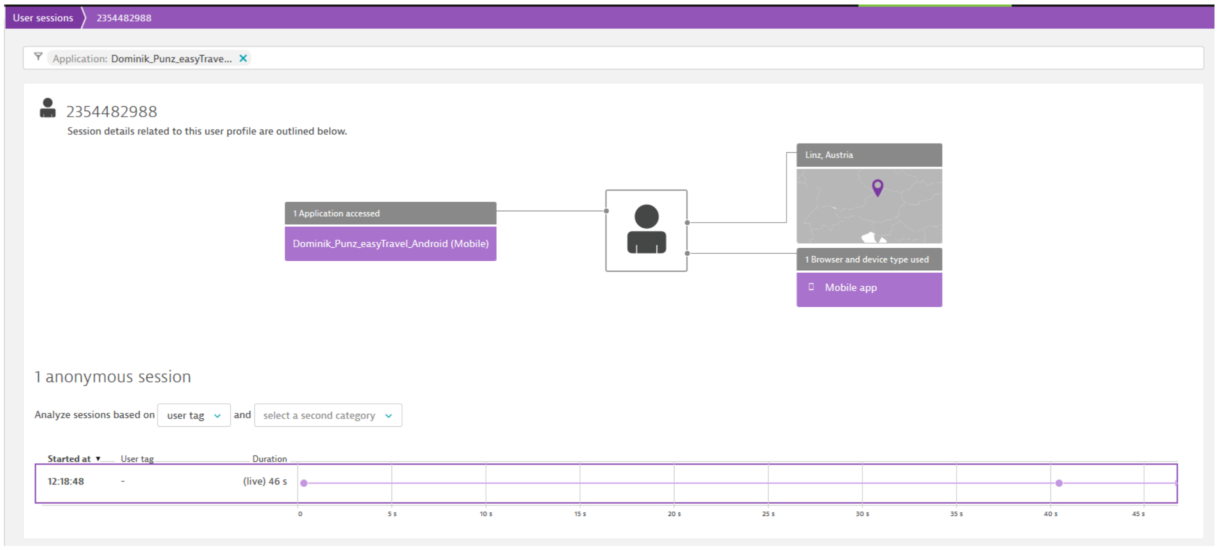
Loading EasyTravel	977 ms
AppStart (MainActivity)	389 ms
Display MainActivity	226 ms
Network Requests	599 ms
/400	599 ms

Lifecycle event: onCreate
Lifecycle event: onStart
Lifecycle event: onResume

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**.

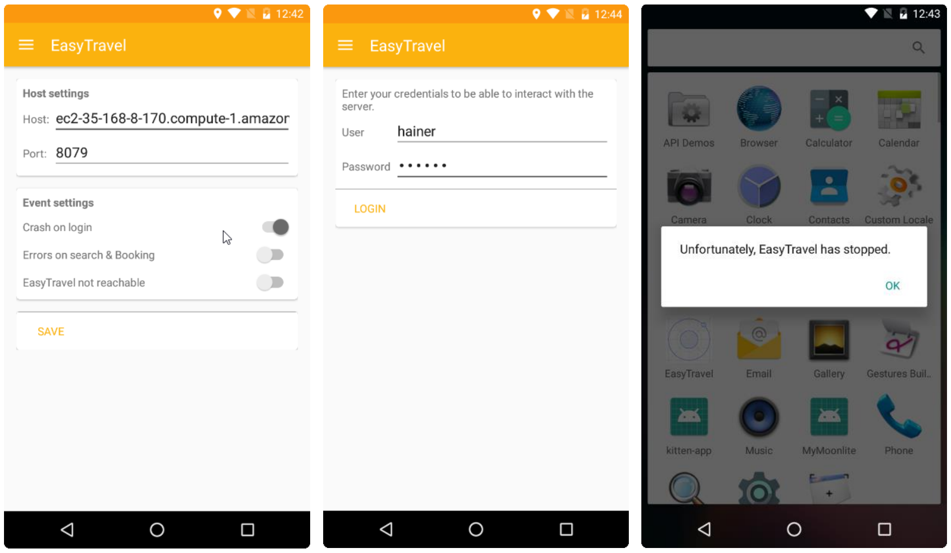

Applications
Dominik_Punz_easyTravel_Android
Service flow
Showing service flow of 'Dominik_Punz_easyTravel_Android' user actions
Today, 11:55 - 12:25 (30 Minutes)
Apply
Show
Response time
Throughput
Add filter
Dominik_Punz_easyTravel_...
Average action duration 350 ms
User actions 11
EasyTravelWebserver:8079
3.7 % response time contribution
easyTravel Customer Fron...
3.6 % response time contribution

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


User sessions
2354482988
Application: Dominik_Punz_easyTrave...
2354482988
Session details related to this user profile are outlined below.
1 Application accessed
Dominik_Punz_easyTravel_Android (Mobile)
Linz, Austria
1 Browser and device type used
Mobile app
1 anonymous session
Analyze sessions based on user tag and select a second category
Started at
User tag
Duration
12:18:48
-
(live) 46 s
0
5 s
10 s
15 s
20 s
25 s
30 s
35 s
40 s
45 s

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**.



12:42
EasyTravel

Host settings
Host: ec2-35-168-8-170.compute-1.amazor
Port: 8079

Event settings
Crash on login
Errors on search & Booking
EasyTravel not reachable

SAVE

12:44
EasyTravel

Enter your credentials to be able to interact with the server.
User hainer
Password ••••••

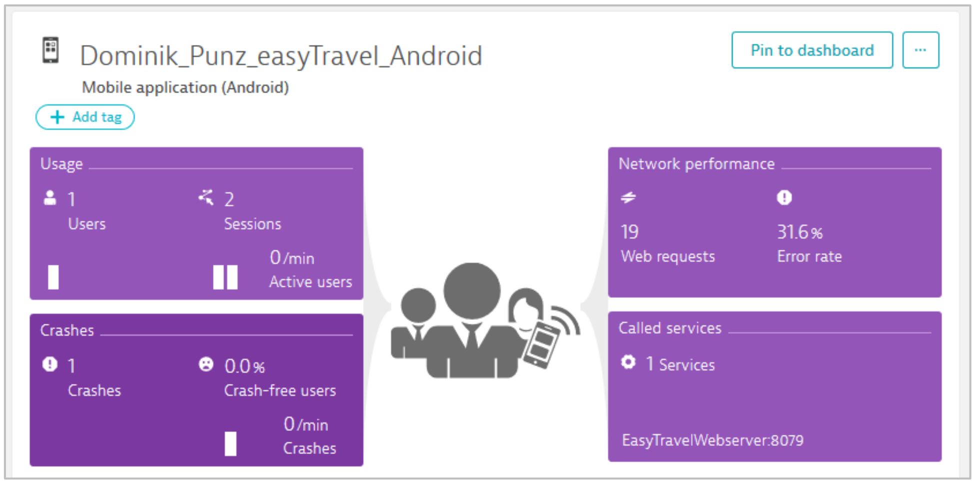
LOGIN

12:43

Unfortunately, EasyTravel has stopped.

OK

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



Dominik_Punz_easyTravel_Android
Mobile application (Android)

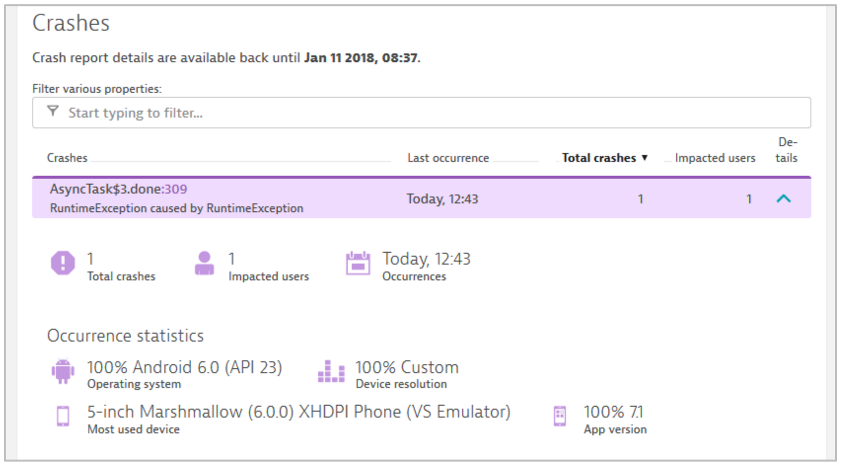
Usage
1 Users
2 Sessions
0 /min Active users

Crashes
1 Crashes
0.0 % Crash-free users
0 /min Crashes

Network performance
19 Web requests
31.6 % Error rate

Called services
1 Services
EasyTravelWebserver:8079

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



Crashes

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

Filter various properties:

Start typing to filter...

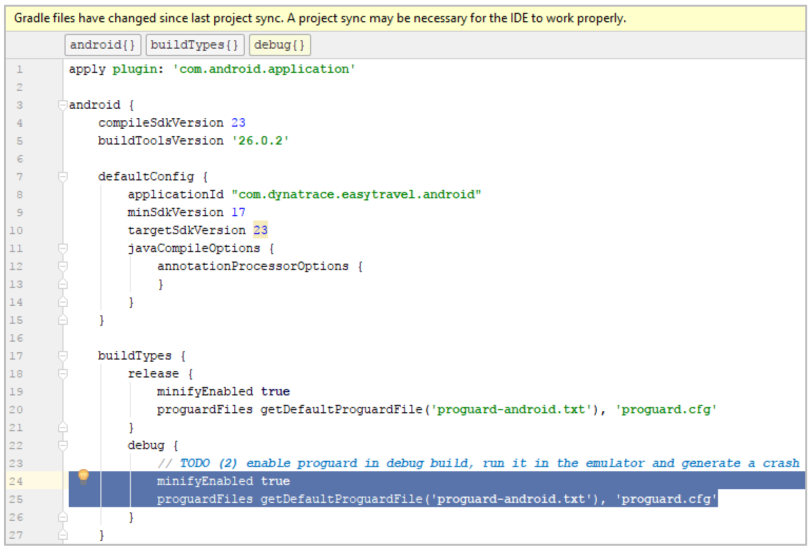
Crashes	Last occurrence	Total crashes ▾	Impacted users	De-tails
AsyncTask$3.done:309 RuntimeException caused by RuntimeException	Today, 12:43	1	1	^

1 Total crashes
1 Impacted users
Today, 12:43 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% 7.1 App version

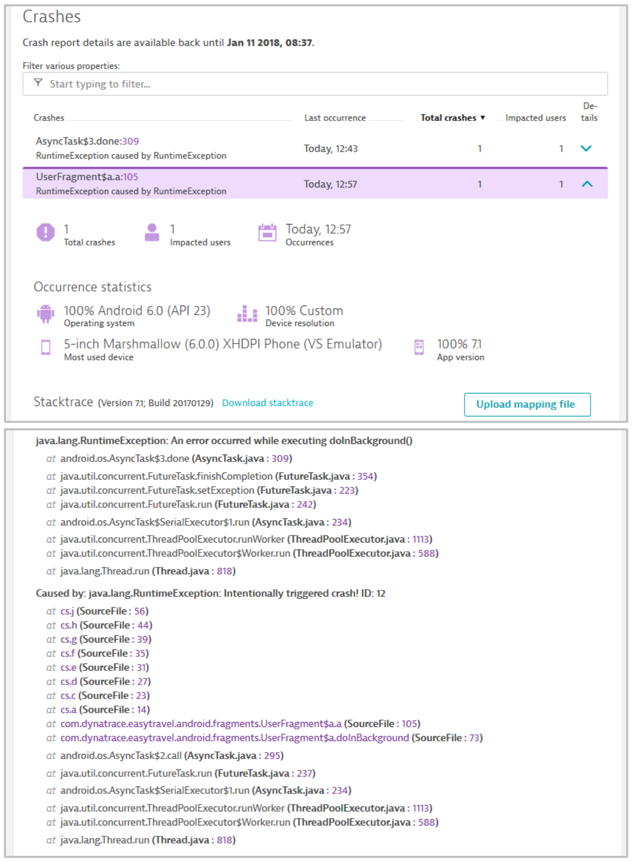
28. Often, mobile applications are obfuscated so the crash stack trace is not readable. Let’s modify the application and use Proguard to obfuscate it. In Android Studio, go to the module gradle script and locate the **TODO (2)**.
29. Enable Proguard in the **debug** build by uncommenting the 2 lines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

```
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:

Crashes	Last occurrence	Total crashes ▼	Impacted users	De-tails
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

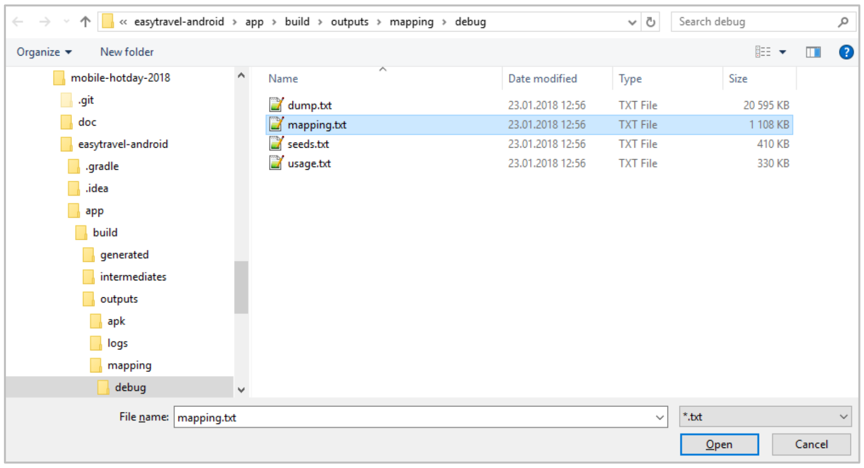
5-inch Marshmallow (6.0.0) XHDPI Phone (VS Emulator) Most used device

100% 7.1 App version

Stacktrace (Version 7.1; Build 20170129) Download stacktrace

```
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)
```

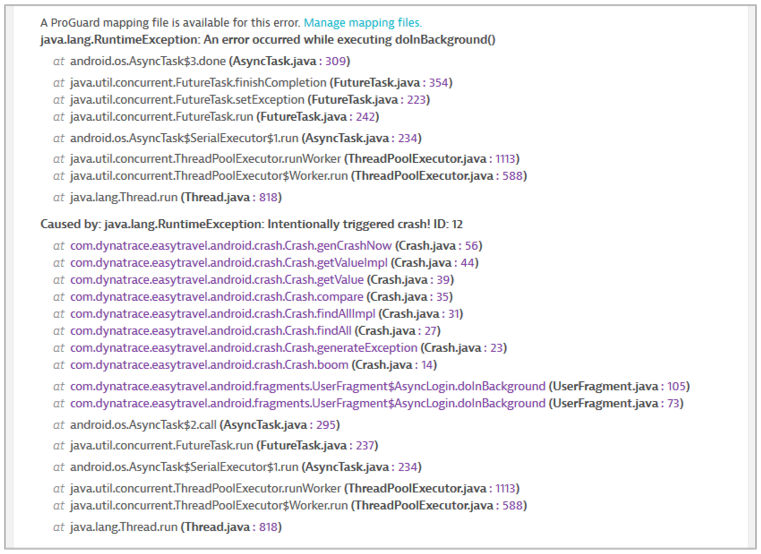
31. Click on **Upload mapping file** and locate the **mapping.txt** file in the **easytravel-android/app/build/outputs/mapping/debug** directory.



Name	Date modified	Type	Size
dump.txt	23.01.2018 12:56	TXT File	20 595 KB
mapping.txt	23.01.2018 12:56	TXT File	1 108 KB
seeds.txt	23.01.2018 12:56	TXT File	410 KB
usage.txt	23.01.2018 12:56	TXT File	330 KB

File name: mapping.txt

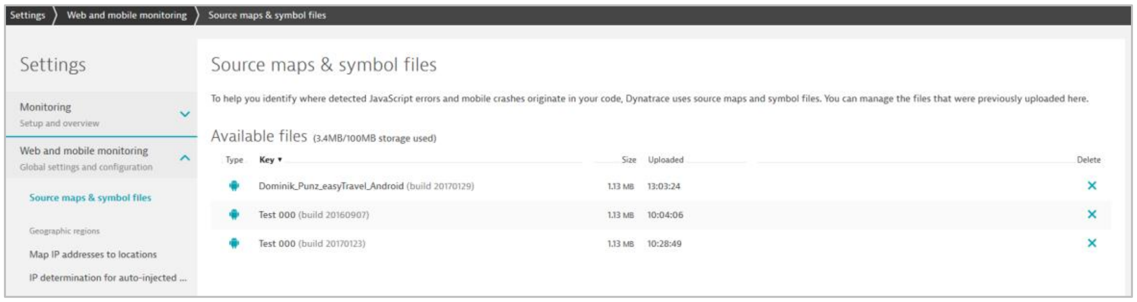
32. View the unobfuscated stack trace in Dynatrace



A ProGuard mapping file is available for this error. Manage mapping files.

```
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 com.dynatrace.easytravel.android.crash.Crash.genCrashNow (Crash.java : 56)
   at com.dynatrace.easytravel.android.crash.Crash.getValueImpl (Crash.java : 44)
   at com.dynatrace.easytravel.android.crash.Crash.getValue (Crash.java : 39)
   at com.dynatrace.easytravel.android.crash.Crash.compare (Crash.java : 35)
   at com.dynatrace.easytravel.android.crash.Crash.findAllImpl (Crash.java : 31)
   at com.dynatrace.easytravel.android.crash.Crash.findAll (Crash.java : 27)
   at com.dynatrace.easytravel.android.crash.Crash.generateException (Crash.java : 23)
   at com.dynatrace.easytravel.android.crash.Crash.boom (Crash.java : 14)
   at com.dynatrace.easytravel.android.fragments.UserFragment$AsyncLogin.doInBackground (UserFragment.java : 105)
   at com.dynatrace.easytravel.android.fragments.UserFragment$AsyncLogin.doInBackground (UserFragment.java : 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)
```

33. View the list of available mapping files


Settings › Web and mobile monitoring › Source maps & symbol files
Settings
Monitoring
Setup and overview
Web and mobile monitoring
Global settings and configuration
Source maps & symbol files
Geographic regions
Map IP addresses to locations
IP determination for auto-injected ...

Source maps & symbol files

To help you identify where detected JavaScript errors and mobile crashes originate in your code, Dynatrace uses source maps and symbol files. You can manage the files that were previously uploaded here.

Available files (3.4MB/100MB storage used)

Type	Key ▾	Size	Uploaded	Delete
	Dominik_Punz_easyTravel_Android (build 20170129)	1.13 MB	13:03:24	✕
	Test 000 (build 20160907)	1.13 MB	10:04:06	✕
	Test 000 (build 20170123)	1.13 MB	10:28:49	✕