SDK Goals

A common good that allows

- Work offline (internal database)
- Communicate with DHIS2 instances
- Facilitate the development of Android apps
Android SDK

Dev's Community

Dhis2 Android SDK

SDK

dhis2

API

Common Good
SDK Today - What does it do?

**Metadata sync** - It replicates all the necessary parts of DHIS2 data model to render entry forms offline across the three data models:

- Program
- ProgramSection
- ProgramStage
- ProgramStageSection
- ProgramStageDataElements
- ProgramIndicators
- ProgramRules
- ProgramAttributes
- ProgramDataElements
- ProgramRelations
- another 40 objects
- TrackedEntity
- TrackedEntitySections
- TrackedEntityAttributes

- Program
- ProgramSection
- ProgramStage
- ProgramStageSection
- ProgramStageDataElements
- ProgramIndicators
- ProgramRules
- ProgramAttributes
- ProgramDataElements
- ProgramRelations
- ...another 40 objects...
- TrackedEntity
- TrackedEntitySections
- TrackedEntityAttributes
SDK Today - Granular data Sync

Data sync synchronizes data with the DHIS2 instance

Parameters:
- Total Number of TEIs/Events (global)
- Number of TEIs/Events per Orgunit
- Number of TEIs/Events per Program

Default behaviour:
1. Organisation unit: Data Capture
2. Prioritize enrolment status: Open
3. Time: based on last update of TEI
SDK Today - What does it do?

**Metadata sync** - all necessary metadata to render forms offline - 50+ tables

**Data sync** - most relevant data

**Data Access Layer** - Access the data stored in the database

```javascript
d2.eventModule().events.uid(eventUid)
```
SDK Today - What does it do?

**Metadata sync** - all necessary metadata to render forms offline

**Data sync** - most relevant data

**Data manager** - Access the data stored in the database

**Expression Evaluation** - Inline

Program Indicators
SDK Today - What does it do?

**Metadata sync** - all necessary metadata to render forms offline

**Data sync** - most relevant data

**Data manager** - Access the data stored in the database

**Expression Evaluation** - Inline Prg Indicators

**Supports for last year DHIS2 versions** - ensures app compatibility with DHIS2 current and two previous versions
SDK Today - What does it do?

**Error management** - Granularity at different levels: tei, event, enrollment, attribute/dataelement

- Show time in error log
SDK Today - What does it do?

**Error management** - Granularity at different levels: tei, event, enrollment, attribute/dataelement
- Show time in error log

**Integrity check** - Missing dependencies
SDK Today - What does it do?

**Error management** - Granularity at different levels: tei, event, enrollment, attribute/dataelement

**Integrity check** - Missing dependencies

**Online search** - teis enrolled in the search org. units
SDK Today - What does it do?

**Error management** - Granularity at different levels: tei, event, enrollment, attribute/dataelement

**Integrity check** - Missing dependencies

**Online search** - teis enrolled in the search org.

**Units**

**Reserved TEI unique attributes** -
Auto-generated to work offline
SDK Today - What does it do?

**Error management** - Granularity at different levels: tei, event, enrollment, attribute/dataelement

**Integrity check** - Missing dependencies

**Online search** - teis enrolled in the search org.

**Units**

**Reserved TEI unique attributes** - Auto-generated to work offline

**Database migrations** - When updating to a new version
SDK - Summary

A common digital good

Accelerates development of DHIS2 offline integrated apps
SDK - Summary

A common digital good

Accelerates development of DHIS2 offline integrated apps

For ANDROID developers
SDK - Summary
A common digital good

Accelerates development of DHIS2 offline integrated apps
For ANDROID developers
Written in JAVA
SDK - Summary
A common digital good

Accelerates development of DHIS2 offline integrated apps
For ANDROID developers
Written in JAVA
Uses SQLite
SDK - Summary

A common digital good

Accelerates development of DHIS2 offline integrated apps

For ANDROID developers

Written in JAVA

Uses SQLite

A group effort - SDK developers + DHIS2 backend team special endpoint to make efficient calls
SDK - Summary

A common digital good

Accelerates development of DHIS2 offline integrated apps

For ANDROID developers

Written in JAVA

Uses SQLite

A group effort - SDK developers + DHIS2 backend team special endpoint to make efficient calls

Open Source - github.com/dhis2/dhis2-android-sdk
Data Access Layer
Data access layer

Stack layer

Public

Private

Repositories

Modules

D2

DB access

API access

DB

API
Data access layer

- D2 is the entry point to interact with the SDK
- For instantiation, we need:
  - Android context (create DB)
  - Server URL
  - App name/version (user-agent)
  - Timeouts
  - Network interceptors
Data access layer

Modules

- Wrapper for related functionality

- It contains:
  - Repositories: one for each entity type (e.g. ProgramRepository)
  - Services and helpers (e.g. PeriodHelper)
Data access layer

Modules

- enrollmentModule
- eventModule
- programModule
- relationshipModule
- trackedEntityModule
- dataSetModule
- dataValueModule
- indicatorModule
- categoryModule
- constantModule
- dataElementModule
- optionModule
- legendSetModule
- organisationUnitModule
- periodModule
- userModule
- smsModule
- maintenanceModule
- systemInfoModule
- systemSettingModule
- importModule
Data access layer

Repositories

- JAVA facade for the DB
- Read for metadata
- Read/Write for data
- They offer a “builder” composition with compile-time validation
- Similar syntax to web API (filters, nested fields, paging)
- Queries in SQL can still be performed
- But general use queries and updates in repositories
Data access layer
Repositories

trackedEntityModule

- trackedEntityTypes
- trackedEntityInstances
- trackedEntityDataValues
- trackedEntityAttributeValues
- trackedEntityAttributes
- trackedEntityTypeAttributes
- trackedEntityInstanceQuery
- reservedValueManager

Repositories

Modules

D2
Data access layer

Schemas

Classes returned by repositories

They may include complex objects

```java
@AutoValue
class Event {

    public String uid();

    public List<TrackedEntityDataValue> trackedEntityDataValues();

    ...}
```
D2Configuration configuration = D2Configuration.builder()
    .appName("app_name")
    .appVersion("1.0.0")
    .context(context)
    .readTimeoutInSeconds(30)
    .connectTimeoutInSeconds(30)
    .writeTimeoutInSeconds(30)
    .networkInterceptors(Collections.singletonList(new StethoInterceptor()))
    .build();

Single<D2> d2Single = D2Manager.setUp(d2Configuration)
    .andThen(D2Manager.setServerUrl(serverUrl))
    .andThen(D2Manager.instantiateD2());

D2 d2 = D2Manager.getD2();
Data access layer
Repositories in detail: fetching data

Fetching event data:

```java
d2
  .eventModule()  (instance)
  .events        (module)
  .getPaged()    (repository)
  .getPaged()    -> LiveData
```
Filter operators depend on value types

d2
  .eventModule()
  .events
  .byOrganisationUnitUid().eq("DiszpKrYNg8")
  .byEventDate().after(new Date("2019-05-05"))
  .get();
Data access layer
Repositories in detail: ordering

Order by EventDate

d2
  .eventModule()
  .events
  .byOrganisationUnitUid().eq("DiszpKrYNg8")
  .byEventDate().after(Date("2019-05-05"))
  .orderByEventDate(DESC)
  .get();
Data access layer
Repositories in detail: nested fields

Only fields stored in entity table are returned by default

d2
  .eventModule()
  .events
  .byOrganisationUnitUid().eq("DiszpKrYNg8")
  .byEventDate().after(Date("2019-05-05"))
  .orderByEventDate(DESC)
  .withTrackedEntityDataValues()
  .get();
Data access layer
Repositories in detail: data upload

Data repositories offer upload capabilities

d2
  .eventModule()
  .events
  .byOrganisationUnitUid().eq("DiszpKrYNg8")
  .byEventDate().after(Date("2019-05-05"))
  .upload();
Data access layer
Repositories in detail: data creation

```java
String eventUid = d2.eventModule().events.add(
    EventCreateProjection.create(
        "enrollent", "program", "programStage", "orgUnit", "attCombo"));

d2.eventModule().events.uid(eventUid).setStatus("COMPLETED");

d2.trackedEntityModule().trackedEntityDataValues
    .value(eventUid, dataElementId).set("5");
    .get();
    .delete();
    .exists();
```
Skeleton app

Starting point for Android apps

What is done:

- SDK dependencies
- D2 initialization
- Login and synchronization
- Some code examples:
  - Display program list
  - Display TEI list

Available on Github
Skeleton app

DEMO
DHIS2 Compatibility

DHIS2 core compatibility

Compatibility with **latest three DHIS 2 core versions** (at least)

Guidelines:

- SDK encapsulates DHIS 2 changes by extending data model
- Sometimes upgrading SDK version to support a new DHIS 2 version implies modifications in code

SDK blocks connection to non-supported DHIS 2 versions

Changes in SDK API will be properly documented in upgrade notes
DHIS2 Compatibility

DHIS2 core version compatibility: compatible changes

```java
class Program {
    ...
    public Boolean captureCoordinates();
    ...
}
```

TRUE

FALSE

POINT

POLYGON

MULTI_POLYGON

NONE

2.29

```java
class Program {
    ...
    @Deprecated
    public Boolean captureCoordinates();
    public FeatureType featureType();
    ...
}
```

2.30
DHIS2 Compatibility

DHIS2 core version compatibility: breaking changes

For example, Relationships in 2.30+

SDK forces to use 2.30+ model. Modifications in code are required

SDK manages compatibility with previous versions
DHIS2 Compatibility

DHIS2 core version compatibility: server upgrades

This approach facilitates server upgrades:

1. Upgrade end-users’ app to next DHIS2 version. They are still compatible with current.
2. Upgrade DHIS2 server. Apps will continue working without interruption.

Diagram:

- **App SDK**
  - 1.0
  - 1.1

- **Server**
  - 2.32
  - 2.33
Error management

Metadata integrity may be tricky

Main guidelines:

- Do not fail, but continue and store the errors
- Keep metadata integrity

Metadata errors are stored for inspection in Maintenance module

- Missing dependencies (FK errors)
- Generic API errors

Data errors are stored to look for solutions (ImportModule)
Performance hints

App / SDK performance is greatly impacted by:

- **Program rules**: they slow down data entry leading to a bad user experience
- **Organisation units**: very large hierarchies cause high synchronization times

The SDK offers some methods to identify potential performance issues
Workflow

1. Initialize D2
2. Login
3. Sync metadata
4. Sync data
5. Upload data
6. WORK
SDK Release

When is this released?

Code available on Github:

https://github.com/dhis2/dhis2-android-sdk
Security and Performance
V1: Architecture, Design and Threat Modelling
V2: Data Storage and Privacy
V3: Cryptography Verification
V4: Authentication and Session Management
V5: Network Communication
V6: Platform Interaction
V7: Code Quality and Build Settings
V8: Resiliency Against Reverse Engineering

<table>
<thead>
<tr>
<th>V1: Architecture, Design and Threat Modelling</th>
<th>P</th>
<th>F</th>
<th>NA</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2: Data Storage and Privacy</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>50,00%</td>
</tr>
<tr>
<td>V3: Cryptography Verification</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>33,33%</td>
</tr>
<tr>
<td>V4: Authentication and Session Management</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>50,00%</td>
</tr>
<tr>
<td>V5: Network Communication</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>83,33%</td>
</tr>
<tr>
<td>V6: Platform Interaction</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>88,89%</td>
</tr>
<tr>
<td>V7: Code Quality and Build Settings</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>9,09%</td>
</tr>
</tbody>
</table>
Performance

/api/programs
/api/programStages
/api/programRules
/api/trackedEntityTypes
/api/relationshipTypes
/api/optionSets
/api/dataSets
/api/dataElements

...
Performance

Testing Methodology

- Simulated environment of multiple users accessing to DHIS2 concurrently via Android SDK
- **Load testing** approach, user count increases over time
- Tests are run multiple times to avoid occasional deviations
- Average values are taken into account as a final result
- Server stats are logged during testing process
- Server is cooled down and restored to its initial state between executions
Performance

Tested System Specs

Hardware
● 64-bit 4-core Intel(R) Xeon(R) Platinum 8175M CPU @ 2.50GHz
● 16GB RAM Memory
● 200GB EBS

Software
● Ubuntu Server 18.04 (64-bit)
● Apache Tomcat 8.5.30
● PostgreSQL 10.6
● DHIS2 2.30 (10.07.2018) / 2.30 (05.07.2019) / 2.32 (05.08.2019)

Data Set
● Sierra Leone demo DB
● Android user
Performance
Tested System Specs

**Tomcat Settings**

**JVM:** `-Xmx8G -Xms8G`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionTimeout</td>
<td>60s</td>
</tr>
<tr>
<td>minSpareThreads</td>
<td>100</td>
</tr>
<tr>
<td>maxThreads</td>
<td>200</td>
</tr>
<tr>
<td>maxConnections</td>
<td>1600</td>
</tr>
<tr>
<td>prestartminSpareThreads</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
Performance
Tested System Specs

**PostgreSQL Settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>300</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>3200 MB</td>
</tr>
<tr>
<td>work_mem</td>
<td>20 MB</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>512 MB</td>
</tr>
<tr>
<td>effective_cache_size</td>
<td>8000 MB</td>
</tr>
<tr>
<td>checkpoint_completion_target</td>
<td>0.8</td>
</tr>
<tr>
<td>synchronous_commit</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Performance
Tested Scenarios

Load Type
➔ Sustained Load: User calls are spread over 60 seconds (users/min)

(Meta)Data Type
➔ Metadata Download: Each user downloads all of its associated metadata
➔ Data Download: Each user downloads all of its associated data
➔ Data Upload: Each user uploads ten new TEIs
Performance

Conventions

**Error:** Every request timed out or not replied with HTTP 200 OK is considered error

**Timeout:** Set to 60 seconds, unless another value is specified for a particular test
Performance - Results

Metadata Download (Sustained)

October 2.30

May 2.30

May 2.32
Performance - Results

Data Download (Sustained)

October 2.30

May 2.30

May 2.32

UiO: University of Oslo
Performance - Results

Data Upload (Sustained)

October 2.30

May 2.30

May 2.32
Performance

Users before failure

Error < 1%

Users

1000
900
800
700
600
500
400
300
200
100
0

Metadata Download Sustained
Data Download Sustained
Data Upload Sustained

October 2.30
May 2.30
May 2.32

100
400
900
60
60
100
20
90
130
Performance

Conclusions

We are very concerned about the performance between Android SDK and DHIS2 API.

There have been big improvements so far:

<table>
<thead>
<tr>
<th>Date</th>
<th>Metadata Download</th>
<th>Data Download</th>
<th>Data Upload</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>May 2.30</td>
<td>300%</td>
<td>0%</td>
<td>350%</td>
</tr>
<tr>
<td>May 2.32</td>
<td>125%</td>
<td>65%</td>
<td>44%</td>
</tr>
</tbody>
</table>

We keep collaborating with DHIS2 core team to improve it.
Roadmap
Roadmap

Documentation (Q2/3)
- Common use cases
- Code examples

SMS synchronization support (Q2/3)
- Shared compression library
- No special metadata configuration
Roadmap

Granular sync (Q3)
- Selective download

Validation rules (aggregated data) (Q3/4)
- It should be a shared library

\[(2 + 2) = 5\]

false
Roadmap

Break the glass support (Q3/4)

- Explicit consent to download a glass-protected TEI
- Implicit consent to upload TEIs
Roadmap

Encrypted database (Q4 -2019)

SQLCipher

- Open source extension of SQLite
- 256-bit AES encryption
- 5% - 15% overhead for encryption
Roadmap

Local Analytics (Q3 - Q4)

A new module for expression engine - **Simple calculation** across data capture Org Units.
Roadmap

Multi server / Multi user (2020)

- A database per user/URL
Roadmap

Widget / UI components (2020)

- Organisation Unit tree
- Datavalue Types
Roadmap

Utility/logic methods/classes
- Enrollment and event status logic
- Validate data entry
- Ping services
- Helpers

Offline / online (2020)

Metadata / Data
Academies
Developers academy

Learn the basics to build your own Android application using the DHIS2 SDK

Good practices on user experience design, testing, debugging using crash statistics and more

**When:** 10 - 13 December 2019

**Where:** Negombo, Sri Lanka

THANKS!!!