

# MVHS Zeiterfassung

*Unterlagen zum Kurs  
„Programme entwickeln für Android Smartphones“  
an der Münchner Volkshochschule*

Lizenz: CC-BY-NC 3.0 (<http://creativecommons.org/licenses/by-nc/3.0/deed.de>)

Ersteller: Dipl.-Ing. (FH) Eugen Richter (WebDucer – IT & Internet Service)

# Inhaltsverzeichnis

Geschichte von Android.....	3
Versionsgeschichte.....	3
Versionen.....	4
Versionsunterschiede (Features).....	4
Versionsverbreitung.....	5
Verlauf der Verbreitung.....	6
Hardware.....	7
Sensoren.....	7
Auflösungen.....	7
Auflösungsverbreitung.....	7
Prozessoren.....	7
Verbreitungswege.....	8
Programmierung.....	9
Lebenszyklus der Activity.....	10
MVHS Zeiterfassungs-App.....	11
Aufgabenstellung.....	11
Tag 1.....	11
Vorgesehene Features / Schritte.....	11
Anlegen eines Android-Projektes.....	11
Layout.....	12
Logik.....	13
Datenbank.....	13
Tag 2.....	13
Vorgesehene Features / Schritte.....	13
SQLite unter Android.....	13
Abfrageoperationen.....	13
Änderungsoperationen.....	14
Prepared Statements.....	14
Tag 3.....	15
Vorgesehene Features / Schritte.....	15
Menüs.....	15
Anlegen einer neuen Activity.....	15
Listen-Ansicht.....	16
Tag 4.....	16
Vorgesehene Features / Schritte.....	16
Übergabe von Parametern an Activities.....	16
Arbeiten mit Date- und Time-Pickern.....	17
Kontextmenü.....	17
Google Market (Developer).....	18
Tag 5.....	21
Vorgesehene Features / Schritte.....	21
Threading.....	21
Dialoge.....	21
Entwicklungstipps.....	22
Vorschläge zur Weiterentwicklung.....	22
Performance Optimierungen.....	24
Anhang.....	25
Installation von Eclipse IDE und ADT.....	25
Installation von Eclipse.....	25
Installation von ADT.....	25
Git – Versionsverwaltung unter Eclipse.....	27
Installation des Plug-ins.....	27
Import eines Projektes.....	28

# Geschichte von Android

Quelle: [http://de.wikipedia.org/wiki/Android\\_\(Betriebssystem\)](http://de.wikipedia.org/wiki/Android_(Betriebssystem))

Stand: 13.04.2011

Die Geschichte von Android beginnt im Jahr 2005 mit der Übernahme der Firma Android von Andy Rubin durch Google. Diese Firma war auf mobile Anwendungen mit Schwerpunkt auf standort-bezogene Dienste. Im Jahr 2007 gründete Google mit weiteren 33 Partnern die Open Handset Alliance und kündigte die Entwicklung eines neuen mobilen Betriebssystems an. Im Oktober 2008 erschien das erste Android Mobiltelefon (HTC Dream / T-Mobile G-1).

Seit dem wurde das Android Betriebssystem weiter verbessert und neue Funktionen erweitert.

## Versionsgeschichte

Version (Name)	Freigabedatum
1.1	10. Februar 2009
1.5 (Cupcake)	30. April 2009
1.6 (Donat)	15. September 2009
2.0 (Eclair)	26. Oktober 2009
2.1 (Eclair)	12. Januar 2010
2.2 (Froyo)	20. Mai 2010
2.3 (Gingerbread)	6. Dezember 2010
2.3.3 (Gingerbread)	23. Februar 2010
3.0 (Honeycomb)	23. Februar 2010 (nur Tablets)
3.1 (Ice Cream)	2011 (Zusammenführung des Codes von 2.3.3 und 3.0)

# Versionen

## Versionsunterschiede (Features)

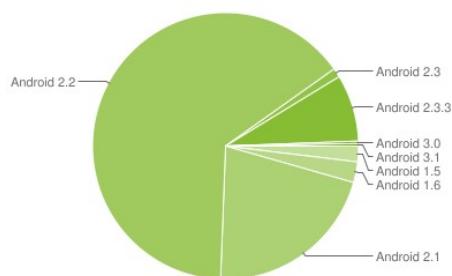
Version (Name)	Datum	Neue Features	Status
1.1	10.02.2009	<ul style="list-style-type: none"><li>• Speichern von MMS-Anhängen</li></ul>	unsupported
1.5 (Cupcake)	30.04.2009	<ul style="list-style-type: none"><li>• <b>Automatischer Wechsel zwischen Hoch- und Querformat</b></li><li>• <b>Bildschirm-Tastatur</b></li><li>• Aufnahme und Wiedergabe von Videos</li><li>• Automatisches Verbinden und Stereo für Bluetooth</li><li>• Weitere Sprachen neben Englisch und Deutsch</li></ul>	Supported
1.6 (Donut)	15.09.2009	<ul style="list-style-type: none"><li>• Virtual Private Networks konfigurierbar</li><li>• <b>Differenzierte Energieverbrauchssteuerung</b></li><li>• Suchfunktion quellenübergreifend und selbstoptimierend</li><li>• Text-to-Speech, Gestenerkennung und mehr als eine Bildschirmauflösung</li></ul>	Supported
2.0 (Eclair)	26.10.2009	<ul style="list-style-type: none"><li>• Digitalzoom und Unterstützung von Blitzlicht</li><li>• <b>Unterstützung von Microsoft Exchange</b></li><li>• Bluetooth 2.1</li></ul>	Supported
2.1 (Eclair)	12.01.2010	<ul style="list-style-type: none"><li>• Animierte Hintergrundbilder</li><li>• Informationen zur Signalstärke</li><li>• <b>Erweiterungen von Webkit (HTML5-Unterstützungen, WebStorage, Geolocation, Video ...)</b></li></ul>	Supported
2.2 (Froyo)	20.05.2010	<ul style="list-style-type: none"><li>• Linux-Kernel 2.6.32, der weniger Arbeitsspeicher benötigt</li><li>• Verwendung von Arbeitsspeicher, der größer als die bisher nutzbaren 256 MByte ist.</li><li>• <b>JIT-Compiler, Erweiterungen für OpenGL ES 2.0, Unterstützung von Flash 10.1</b></li><li>• <b>Tethering</b></li><li>• <b>Speicherbarkeit von Apps auf der SD-Karte (App2SD)</b></li><li>• Android Cloud to Device Messaging Framework: Möglichkeit PUSH in die eigenen Anwendungen zu implementieren.</li><li>• Bluetooth-Sprachwahl</li></ul>	Supported
2.3 (Gingerbread)	06.12.2010	<ul style="list-style-type: none"><li>• Linux-Kernel 2.6.35.7</li><li>• <b>Unterstützung von WebM</b></li><li>• <b>Unterstützung von HTML5 Audio</b></li><li>• Unterstützung von Google TV</li><li>• <b>Unterstützung von Near Field Communication</b></li></ul>	Supported

Version (Name)	Datum	Neue Features	Status
		<ul style="list-style-type: none"> <li>• <b>Parallele Garbage Collection für ruckelfreie Animationen</b></li> <li>• Verbesserte Integration von sozialen Netzwerken</li> <li>• <b>Unterstützung von Gyroskopen (nicht zu verwechseln mit Bewegungssensoren) und anderen Sensoren (u.a. Barometer, Schwerekraftsensor)</b></li> <li>• <b>Integrierter SIP-Client für VoIP</b></li> <li>• Integrierter Downloadmanager</li> <li>• <b>Unterstützung des Ext4-Dateisystems</b></li> </ul>	
2.3.3 (Gingerbread)	23.02.2011	<ul style="list-style-type: none"> <li>• <b>Dual-Core-Unterstützung</b></li> <li>• <b>Unterstützung von Dual-Core-Apps auf Single-Core-Geräten</b></li> <li>• Verbesserte Unterstützung der NFC-Technik</li> <li>• Verbesserte Bluetooth-Unterstützung</li> <li>• Kleinere Verbesserungen</li> </ul>	Current
3.0 (Honeycomb)	23.02.2011	<ul style="list-style-type: none"> <li>• <b>Benutzerfreundlichere Oberfläche</b></li> <li>• <b>Verbesserte Unterstützung für Tablet-Computer</b></li> <li>• Google Talk mit Videotelefonie</li> <li>• Neue Browser-Funktionen: Synchronisierung der Lesezeichen mit Google Chrome, Tabs, automatisches Ausfüllen von Formularen und Inkognito-Modus beim Surfen</li> </ul>	Current
3.1 (Ice Cream)	2011	<ul style="list-style-type: none"> <li>• Zusammenführung der Entwicklungslinien 2.x und 3.x und Google TV</li> </ul>	Future

## Versionsverbreitung

Quelle: <http://developer.android.com/resources/dashboard/platform-versions.html>

Stand: 1. Juni 2011

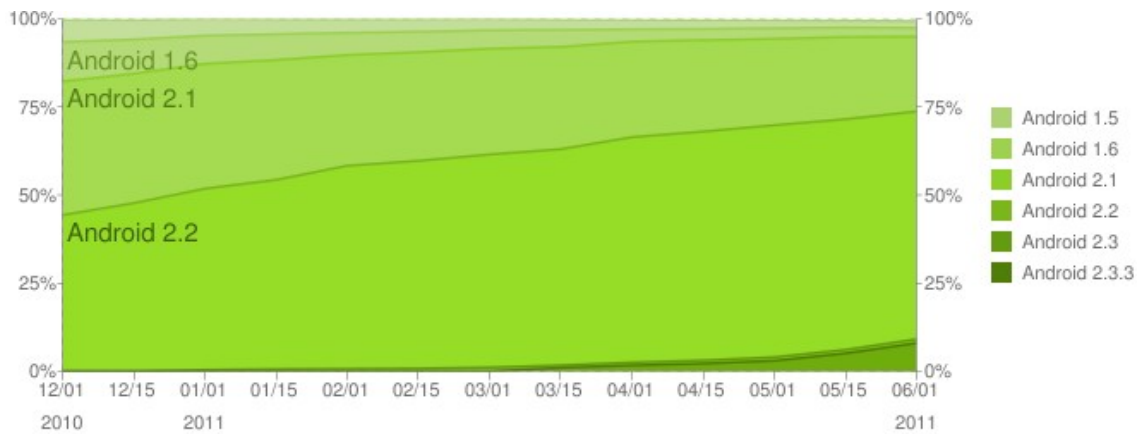


Unter <http://developer.android.com/resources/dashboard/platform-versions.html> liefert Google Statistiken zur Verbreitung der unterschiedlichen Versionen. Die Daten werden durch die Zugriffe auf Android Market zusammengetragen. Somit beinhaltet diese Statistik keine Geräte, die keinen Zugriff auf Android Market haben (z.B.: 1&1 Tablet, der Zugriff nur auf 1&1 AppStore hat und weitere günstige Geräte).

Version	API Level	Verbreitung
Android 1.5	3	1.90%
Android 1.6	4	2,50%
Android 2.1	7	21,20%
Android 2.2	8	64,60%

Version	API Level	Verbreitung
Android 2.3	9	1,10%
Android 2.3.3	10	8,10%
Android 3.0	11	0,30%
Android 3.1	12	0,30%

## Verlauf der Verbreitung



# Hardware

## Sensoren

- GPS
- Kompass
- Gyroskop
- Beschleunigungssensor
- Lagesensor
- NFC
- Annäherungssensor
- Kamera
- Mikrofone

## Auflösungen

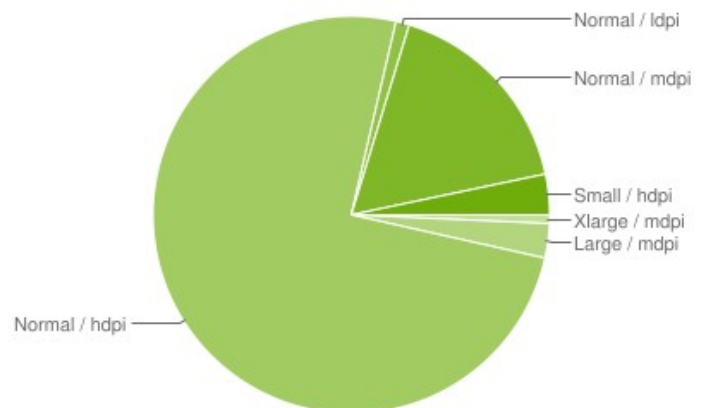
- QVGA (320 x 240) [Low Desitiy]
- HVGA (320 x 480) [Midle Density]
- WVGA (800 x 480) [Hight Density]
- WXGA (1024 x 600 / 1280 x 800) [High Density] Tablets

## Auflösungsverbreitung

Quelle: <http://developer.android.com/resources/dashboard/screens.html>

Stand: 01.Juni 2011

	ldpi	mdpi	hdpi	xhdpi
<b>Small</b>			3,30%	
<b>Normal</b>	1,10%	17,00%	75,20%	
<b>Large</b>		2,80%		
<b>xLarge</b>		0,70%		



## Prozessoren

- ARM Prozessoren mit 400 – 1GHz (Single Core)
- ARM Prozessoren mit 600 – 1,6 GHz (Dual Core)

## Verbreitungswege

Die Android Apps können im Gegensatz zu Apple Apps von jeder Quelle bezogen werden. Es gibt keine Pflicht das Google Market zu benutzen. Aus diesem Grund etablieren sich langsam auch alternative Markets, die unter anderem andere Vergütungsmodelle für Programmierer und andere Bezahlmethoden anbieten.

Weiterhin bieten einige Smartphone Hersteller (entweder nur diesen oder parallel zu Google Market) herstellerspezifische Markets (z.B.: für Samsung Galaxy Tab).

Folgende Bezugsquellen für Apps sind möglich (kein Anspruch auf Vollständigkeit):

- Google Market
- Amazon AppStore (nur USA momentan)
- Androidpit
- Entwickler-Homepage
- Eigener PC/Mac

Bei allen Quellen, außer natürlich Google Market, muss auf dem Android Gerät der Bezug von Apps aus unbekannten Quellen aktiviert sein (Einstellungen → Anwendungen).

Bei den kostenpflichtigen Apps bekommt der Entwickler (bei Google und Amazon) 70% der Einnahmen und 30% der Betreiber des Markets.

Weitere Verdienstmöglichkeiten ergeben sich durch die Werbeeinblendungen in den Apps (z.B. durch AddMobile – gehört mittlerweile auch zu Google).



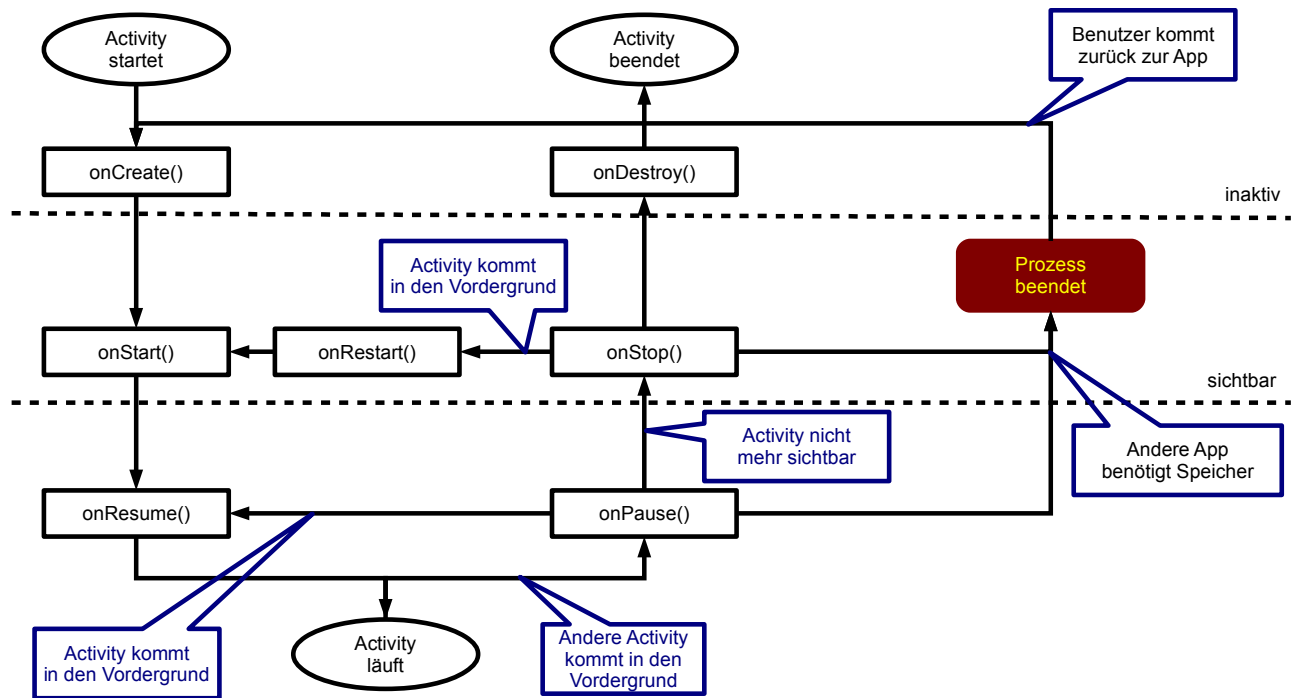
## Programmierung

Die Entwicklung von Android Apps erfolgt grundsätzlich in Java. Android unterstützt dabei Java 1.5 komplett und das Android SDK liefert zusätzlich noch spezielle Bibliotheken, die an das mobile System angepasst wurden (Bluetooth, JSON, XML.Sax, JUnit, W3C.DOM). Die Apps laufen auf dem Gerät (und im Emulator) in einer von anderen Apps abgeschotteten Sandbox. Die eingesetzte JVM wurde von Google neu entwickelt und an die Architektur eines mobilen Systems mit wenig Ressourcen angepasst. Die JVM heißt unter Android Dalvik VM diese kann keinen nativen Java-Code ausführen, genauso wenig wie die Oracle JVM den Dalvik Byte-Code nicht ausführen kann. Beim Kompilieren eines Android Projektes wird aus dem Java-Byte-Code (Kellerautomat) ein Dalvik-Byte-Code (Registerautomat) erstellt.

Nur Performance lastige Anteile wurden in Android in C/C++ erstellt (Treiber, Mediacodecs, Webbrowser Engine, SQLite und OpenGL). Diese Möglichkeit ist auch für die Entwicklung eigener Apps gegeben, und sollte nur dann verwendet werden, wenn es Performance-Engpässe gibt. Da für C/C++ Code kein Garbage Collector vorhanden ist, muss der Programmierer sich selbst um die Freigabe der Ressourcen kümmern.

Neben den eigentlichen Apps sind auch die sogenannten Widgets möglich. Diese können einen schnellen Zugriff auf die Daten Ihrer App gewähren, ohne dass der Benutzer die komplette App starten muss. Eine gut durchdachte Erweiterung einer App um ein oder mehrere Widgets kann den Ausschlag zur Konkurrenz geben.

# Lebenszyklus der Activity



# MVHS Zeiterfassungs-App

## Aufgabenstellung

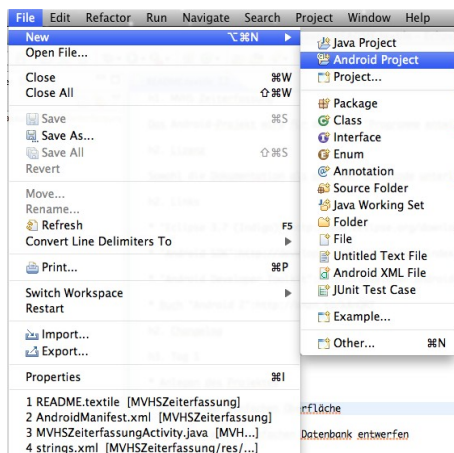
Es soll eine App für die Arbeitszeiterfassung erstellt werden, die auf den Android-Geräten ab Version 1.6 lauffähig ist. Die App soll in mehreren Stufen durch das Hinzufügen von neuen Features aufgebaut werden.

## Tag 1

### Vorgesehene Features / Schritte

- Anlegen eines neuen Android-Projektes
- Erstellung des Layouts für die Zeiterfassung
- Erstellung der Logik für die Zeiterfassung
- Planung der Datenbank (mithilfe der Firefox-Erweiterung SQLite Manager)

### Anlegen eines Android-Projektes



Neues Android-Projekt wird über das Menü „File“ → „New“ → „Android Project“ angelegt (Abb. 1). Sollte dieser Menüpunkt nicht sichtbar sein, gehen Sie auf „Others“ und wählen Sie dann im folgenden Dialog „Android Project“.

Im folgenden Assistenten müssen folgende Angaben gemacht werden (Abb. 3 und 4):

- Project name (Projektname ohne Sonderzeichen)
- Build Target (Compiler-Version, mit der die App kompiliert wird)
- Application Name (Bezeichnung der App, so wie diese im Menü des Smartphones erscheinen soll)

Abb. 1: Neues Android Projekt

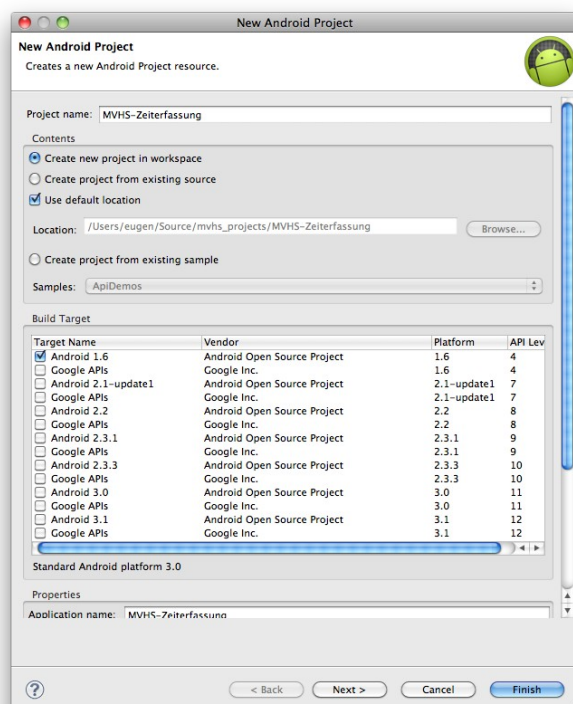


Abb. 2: Angaben für neues Projekt (1)

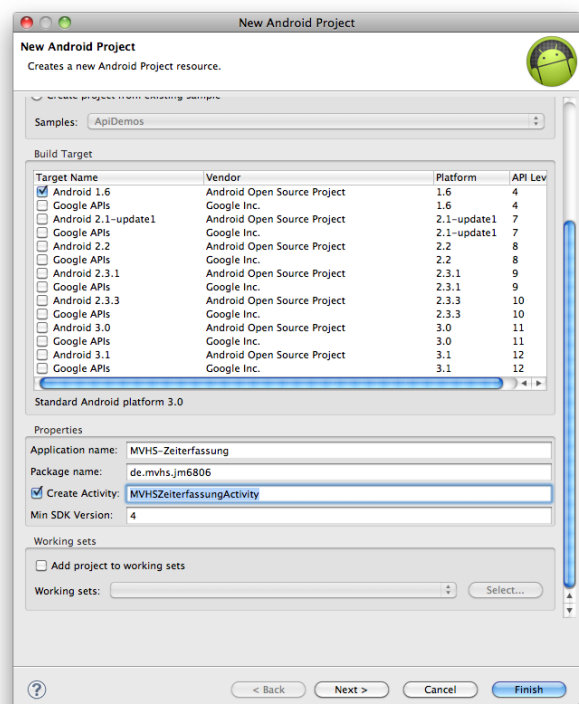


Abb. 3: Angaben für neues Projekt (2)

- Package name (Hierarchische Bezeichnung des Java-Packages. Of als umgekehrte URL angegeben)

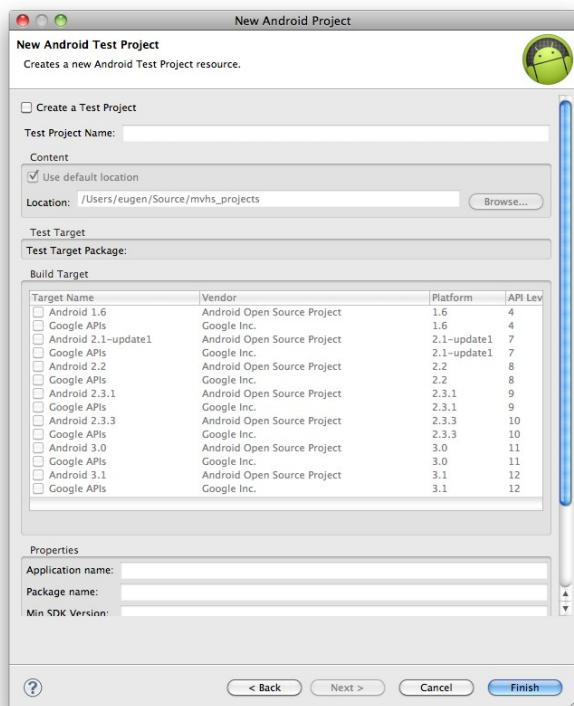


Abb. 4: JUnit Test-Project

## Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:text="@string/start_time"/>
    <EditText
        android:id="@+id/txtStartTime" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </EditText>
    <TextView
        android:text="@string/end_time" android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>
    <EditText
        android:id="@+id/txtEndTime" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </EditText>
    <Button
        android:id="@+id/btnStart" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/button_start"
        android:onClick="onButtonClick">
    </Button>
    <Button
        android:id="@+id/btnEnd" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/button_end"
        android:onClick="onButtonClick">
    </Button>
</LinearLayout>
```

- Min SDK version (minimale Anforderungen an das Gerät)

Für komplexere

Apps ist es

empfehlenswert

zur App auch

TestUnits zu

schreiben, um

bei neueren

Versionen die

Funktionalität

automatisiert

testen zu

können. Das

kann im nach-

folgenden

Dialog mit-

erledigt werden

(Abb. 4).

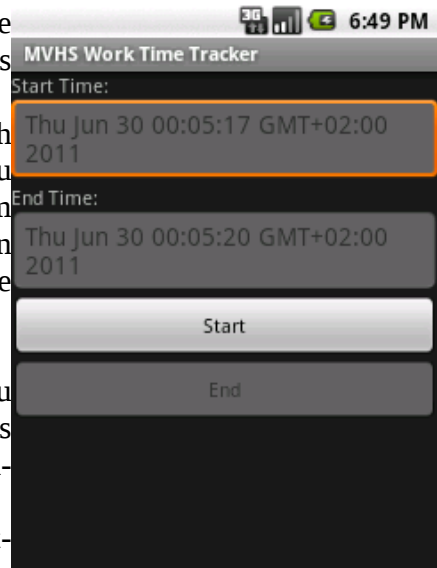


Abb. 5: Layout

## Logik

In der ersten Entwicklungsphase soll beim Betätigen des Knopfes „Start“ die aktuelle Zeit in das erste Textfeld ausgegeben werden und der „End“-Knopf aktiviert werden. Beim Betätigen des Knopfes „End“ soll die aktuelle Zeit in das zweite Textfeld ausgegeben werden und der „Start“-Knopf wieder aktiviert werden.

## Datenbank

Feldbezeichnung	Typ	Beschreibung
<code>_id</code>	Integer	Primärschlüssel des Datensatzes
<code>start_time</code>	Text	Startdatum des Datensatzes in Form „YYYY-MM-DD HH:mm:ss“ (z.B.: 2011-06-29 19:03:53)
<code>end_time</code>	Text	Enddatum des Datensatzes in Form „YYYY-MM-DD HH:mm:ss“ (z.B.: 2011-06-29 19:23:12)

## Tag 2

### Vorgesehene Features / Schritte

- Einführung in SQLite
- Implementierung der DB-Schnittstelle
- Persistente Zeiterfassung (Speichern der Daten in der DB)
- Zugriff auf die Daten auf den Emulator / Smartphone über DDMS-Perspektive

## SQLite unter Android

Für den Zugriff auf die SQLite-Datenbank steht unter Android die abstrakte Klasse `SQLiteOpenHelper` zur Verfügung. Diese Klasse regelt mit den Methoden `onCreate` und `onUpdate` die Verwaltungsmethoden die Erzeugung und Aktualisierung der Datenbank.

Die SQL-Anweisungen können mit dem DB-Parameter aus dem Helper (`SQLiteOpenHelper.getReadableDatabase()` oder `SQLiteOpenHelper.getWritableDatabase()`) auf vier Arten durchgeführt werden.

1. `db.rawQuery`: Als Parameter wird der auszuführende SQL-String übergeben.
2. `db.query`: Als Parameter werden dabei die Bestandteile der Abfrage übergeben.
3. Prepared Statements: Vorkompilierte SQL Anweisungen (sehr performant im Vergleich zu den beiden anderen Methoden).
4. Spezielle Änderungsfunktionen (`update`, `delete`, `insert`).

### Abfrageoperationen

SQLite rawQuery	SQLite query
<pre>SELECT     _id,     start_time,     end_time FROM</pre>	<pre>db.query(     false, // distinct?     "time_tracking" // From Tabelle     new String[] { // Select Spalten         "_id",</pre>

<pre> time_tracking WHERE _id &lt; 100 ORDER BY start_time DESC </pre>	<pre> "start_time", "end_time"}, _id &lt; ", // Where-Bedingung new String[] { // Parameter für where "100"}, null, // Group By Anweisung null, // Having Anweisung "start_time DESC", // Order By null); // Limit </pre>
--	---

## Änderungsoperationen

Operation	SQLite rowQuery	Spezialfunktion
Insert	<pre> INSERT INTO time_tracking (start_time, end_time) VALUES ('2011-07-06 08:00:00', '2011-07-06 17:00:00') </pre>	<pre> Content Values werte = new ContentValues(); werte.put( "start_time", "2011-07-06 08:00:00"); werte.put( "end_time", "2011-07-06 17:00:00"); db.insert( "time_tracking", null, werte); </pre>
Update	<pre> UPDATE time_tracking SET start_time = '2011-07-06 07:15:00', end_time = '2011-07-07 17:45:00' WHERE _id = 100 </pre>	<pre> ContentValues werte = new ContentValues(); werte.put( "start_time", "2011-07-06 07:15:00"); werte.put( "end_time", "2011-07-06 17:45:00"); db.update( "time_tracking", werte, "_id=?", new String[]{"100"}); </pre>
Delete	<pre> DELETE FROM time_tracking WHERE _id = 100 </pre>	<pre> db.delete( "time_tracking", "_id=?", new String[]{"100"}); </pre>

## Prepared Statements

Die Statements liefern als Ergebnis immer 1 x 1 Zelle. Somit sind diese nicht für „normale“ Abfragen geeignet, aber für spezielle Aufgaben, die nur einen einzigen Wert (z.B. „SELECT count(\*) ...“) zurückliefern oder gar kein Ergebnis notwendig ist (Update, Delete, Insert). Diese Statements werden beim Kompilieren des Projektes für SQLite kompiliert und erreichen dadurch sehr hohe Performance (ca. 4 x schneller als rowQuery Anweisungen).

Mögliche Statements sind:

- `execute()`: nicht für SELECT, UPDATE, INSERT, DELETE, aber für CREATE, DROP (Tabelle, View, Index) geeignet.
- `executeInsert()`: für INSERT Anweisungen geeignet.
- `executeUpdateDelete()`: Erst ab Android 2.2 verfügbar und für UPDATE und DELETE Anweisungen geeignet.
- `simpleQueryForLong()`: liefert einen Integer in einer 1 x 1 Tabelle als Ergebnis. Gut für „SELECT count(\*) FROM tabel“ geeignet.

- `simpleQueryForString()`: Arbeitet genauso wie `simpleQueryForLong`, liefert aber statt Integer einen Strings-Wert zurück.

### Beispiel

```
SQLiteStatement stmInsert = db.compileStatement(
    "INSERT INTO time_tracking " +
    "(_id, start_time, end_time) " +
    "VALUES (?, ?, ?)");
stmInsert.bindLong(1, 1001);
stmInsert.bindString(2, "2011-07-06 07:00:00");
stmInsert.bindString(3, "2011-07-06 17:00:00");
long id = stmInsert.executeInsert();
```

## Tag 3

### Vorgesehene Features / Schritte

- Bereinigung bei "Start"-Drücken
- Lokales Datumsformat bei der Anzeige
- Listenansicht der gespeicherten Daten
- Menü
  - Schließen
  - Auflistung
  - Zurücksetzen

### Menüs

Unter Android gibt es zwei Menü-Typen.

5. Options-Menü: Dieses erscheint beim Betätigen der „Menü-Taste“ auf dem Gerät und wird beim Start der Activity nur einmal erzeugt. Es werden fünf erste Einträge des Menüs direkt angezeigt. Ab sechs Elementen wird an der 6. Stelle „mehr ...“ angezeigt. Erst nach dem Klick auf diesen Eintrag, werden die Einträge 6 – X als scrollbare Liste angezeigt.
6. Kontext-Menü: Dieses wird jedes Mal direkt vor dem Anzeigen erzeugt. Dadurch ist es möglich, abhängig von dem geklickten Element, ein anderes Kontext-Menü zu erzeugen. Angezeigt wird dieses als scrollbare Liste von Einträgen. Kontext-Menü unterstützt keine Bilder.

Das Einbinden eines Optionsmenüs erfolgt über das Überschreiben der Methode `onOptionsItemSelected()`. Die Auswertung des gewählten Eintrags erfolgt über die überschriebene Methode der Activity `onOptionsItemSelected()`.

### Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="@string/opt_clear" android:id="@+id/opt_clear"></item>
    <item android:title="@string/opt_list_records" android:id="@+id/opt_list"></item>
    <item android:title="@string/opt_close" android:id="@+id/opt_close"></item>
</menu>
```

### Anlegen einer neuen Activity

- Anlegen einer Klasse, die von einer der Activity-Klassen erbt (Activity, ListActivity ...). Die Klasse nach eigenen Vorstellungen ausprogrammieren.

```
public class RecordList extends ListActivity { ... }
```

- Anlegen einer Layout-Datei für diese Activity (im Verzeichnis „res/layout“).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ListView
        android:id="@+id/android:list"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent">
    </ListView>
</LinearLayout>
```

- Bekanntmachung der Activity für die App in der Android-Manifest-Datei.

```
<activity android:name=".RecordList" android:label="@string/record_list"/>
```

- Definition einer Aufrufstelle (Button, Option- oder Kontext-Menü).

```
final Intent listRecordActivity = new Intent(this, RecordList.class);
startActivity(listRecordActivity);
```

## Listen-Ansicht

Für eine Listenansicht erbt die Activity-Klasse statt von der Activity, von der ListActivity. Diese bietet zusätzliche Funktionalität, um mit Listen umzugehen. In der Layout-Datei muss dabei einer Liste mit der id „android:list“ vorhanden sein. Der SimpleCursorAdapter (oder eine eigene Adapter-Klasse) steuert den Cursor der DB-Ergebnisse und definiert, in welchen Element eines Listeneintrages eine bestimmte Spalte angezeigt werden soll. Der Cursor des DB-Ergebnisses muss dabei immer die „\_id“-Spalte mitführen.

Mit dem Aufruf der ListActivity-Methode „setListAdapter“ mit dem Adapter als Parameter wird der Liste der Activity der Adapter zugewiesen und die Liste somit gefüllt.

## Tag 4

### Vorgesehene Features / Schritte

- Bearbeitungs-Fenster
- Kontext-Menü
  - Delete
  - Edit

## Übergabe von Parametern an Activities

Es kommt immer wieder vor, dass beim Wechsel einer Activity an diese bestimmte Werte übergeben werden müssen. In unserem Fall muss an das Bearbeitungsdialog die ID des Datensatzes übergeben werden. Das wird bei der Initialisierung des Intent über ein **PUT** Befehl übergeben.

```
Intent intentEdit = new Intent(this, EditRecord.class);
intentEdit.putExtra("id", id);
startActivity(intentEdit);
```

Im Bearbeitungsdialog muss dieses EXTRA noch entgegengenommen und verwendet werden.

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
    _ID = extras.getLong("id");
}
```



## Arbeiten mit Date- und Time-Pickern

Die Picker nehmen keine Date-Objekte entgegen, sondern müssen mit den Integer-Werten initialisiert werden. Bei DatePicker muss dabei beachtet werden, dass zum .getYear() aus dem Date-Objekt noch 1900 addiert werden müssen (und beim Auslesen wieder subtrahiert).

```
dpEnd.init(
    endDate.getYear() + 1900,
    endDate.getMonth(),
    endDate.getDate(),
    null);
tpEnd.setCurrentHour(endDate.getHours());
tpEnd.setCurrentMinute(endDate.getMinutes());

Date endDate = new Date(
    dpEnd.getYear() - 1900,
    dpEnd.getMonth(),
    dpEnd.getDayOfMonth(),
    tpEnd.getCurrentHour(),
    tpEnd.getCurrentMinute());
```

## Kontextmenü

Das Kontextmenü wird in der folgenden Methode initialisiert. Das Menü selbst wird, wie auch das Optionsmenü, über XML definiert, und kann für beide Menü-Möglichkeiten verwendet werden.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.context_menu, menu);

    super.onCreateContextMenu(menu, v, menuInfo);
}
```

In der onCreate Methode müssen alle Elemente registrier werden, die Kontext-Menü unterstützen sollten.

```
registerForContextMenu(findViewById(android.R.id.list));
```

In unserem Fall ist es die Liste. Bei Auswerten des Kontextmenüs benötigen wir noch Information, welcher der Listenelemente das Kontextmenü ausgelöst hat. Das erfolgt über die Abfrage des Adapters (wir benötigen hier die ID des DB-Datensatzes).

```
@Override
public boolean onContextItemSelected(MenuItem item) {

    AdapterView.AdapterContextMenuInfo info =
        (AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
    long id = info.id;
    _Position = info.position;

    switch (item.getItemId()) {
        case R.id.ctx_delete:
            DBHelper dbHelper = new DBHelper(this);
            SQLiteDatabase db = dbHelper.getWritableDatabase();
            db.delete(
                TimeTrackingTable.TABLE_NAME,
                TimeTrackingTable.ID + "=?",
                new String[]{String.valueOf(id)});
            db.close();

            LoadData();
            break;

        case R.id.ctx_edit:
            Intent intentEdit = new Intent(this, EditRecord.class);
            intentEdit.putExtra("id", id);
            startActivity(intentEdit);

            break;
    }
```

```

default:
    break;
}

return super.onContextItemSelected(item);
}

```

## Google Market (Developer)


webducer@gmail.com | [Startseite](#) | [Hilfe](#) | [Android.com](#) | [Abmelden](#)

---

**WebDucer**  
 android@webducer.de  
[Profil » bearbeiten](#)

**Alle Android Market-Einträge**



[Worktime Tracker \(WD\)](#) 0.0.5 alpha  
 Apps: Produktivität  
[In-App-Produkte](#)

(1) ★★★★★  
[Kommentare](#)

210 Installationen insgesamt  
 111 aktive Installationen (52 %)  
[Statistiken](#)

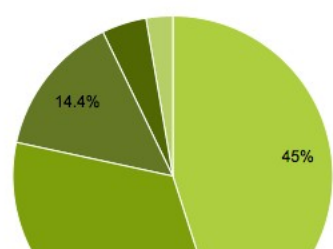
Kostenlos [Fehler](#)

✓ Veröffentlicht  
[Für diese App werben](#)



Abb. 6: Übersicht der Apps des Entwicklers

**Attributaufschlüsselung bis zum 2. August 2011**

**Plattform-Version**


de.webducer.android.worktime	
1	Android 2.2 45,0% (50)
2	Android 2.3.3 33,3% (37)
3	Android 2.1 14,4% (16)
4	Android 1.6 4,5% (5)
5	
6	
7	
8	
9	

Alle Apps in meiner Kategorie	
Android 2.2	52,4%
Android 2.3.3	18,7%
Android 2.1	17,1%
Android 1.6	3,1%
Android 1.5	2,2%
Android 2.3	0,4%
Android 3.0	0,4%
Android 2.0.1	0,1%
Android 1.0	0,0%

**Land**

de.webducer.android.worktime	
1	Deutschland 58,6% (65)


**Anwendung bearbeiten**

[Produktdetails](#)
[APK-Dateien](#)

[Veröffentlichung aufheben](#)
[Speichern](#)

[Zum Basismodus wechseln »](#)

**Aktiv**  
 Wenn Sie auf "Speichern" klicken, werden diese APKs im Android Market live geschaltet.
 



Versionscode: 5  
 Versionsname: 0.0.5 alpha  
 Größe: 124k  
 Lokalisiert für: Standard, German, Russian  
 Berechtigungen:  
 android.permission.WRITE\_EXTERNAL\_STORAGE  
 Funktionen:  
 android.hardware.touchscreen  
[« Weniger](#)


API-Ebene: 4-12+  
 Unterstützte Bildschirme: small-xlarge  
 OpenGL-Texturen: Alle

[Deaktivieren](#)

**Neu**  

[APK hochladen](#)

**Zuvor aktiv**  



Versionscode: 4  
 Versionsname: 0.0.4 alpha  
 Größe: 98k  
 Lokalisiert für: Standard, German, Russian  
 Berechtigungen:  
 android.permission.WRITE\_EXTERNAL\_STORAGE  
 Funktionen:  
 android.hardware.touchscreen  
[« Weniger](#)

API-Ebene: 4-12+  
 Unterstützte Bildschirme: small-xlarge  
 OpenGL-Texturen: Alle

[Reaktivieren](#)

Abb. 9: Unterschiedliche Versionen der App (VersionCode)

Es ist auch möglich dieselbe App für unterschiedliche Versionen zu optimieren und mehrere apk-Dateien zur Verfügung zu stellen (z.B. Tablet Edition und Smartphone Edition).

## Anwendung bearbeiten

**Produktdetails**


**APK-Dateien**

**Veröffentlichung aufheben**

**Speichern**


**Inhalte hochladen**

**Screenshots**  
mindestens 2  
[Weiteren hinzufügen](#)



**Screenshots:**  
320 x 480, 480 x 800,  
480 x 854, 1280 x 800  
24-Bit-PNG oder -JPEG (kein Alpha)  
Randlos, kein Rahmen im Bild  
Sie können Screenshots im Querformat hochladen. Die Miniaturansichten sehen gedreht aus, aber die eigentlichen Bilder und ihre jeweilige Ausrichtung bleiben erhalten.

**Hochauflösendes Symbol für App**  
[Weitere Informationen](#)



**Hochauflösendes Symbol für App:**  
512 x 512  
32-Bit-PNG oder -JPEG  
Maximale Größe: 1024 KB

Abb. 10: Informationen zur App (1)

## Liste der Details

**Sprache** | \*English (en) | [русский \(ru\)](#) | [Deutsch \(de\)](#) |  
[Sprache hinzufügen](#) Das Sternsymbol (\*) weist auf die Standardsprache hin.

☒ Den Eintrag auf Englisch entfernen

**Title (English)**   
 21 Zeichen (maximal 30)

**Description (English)**   
 Features  
 =====  
 - Track working time  
 - Show list of tracked working times (records)  
 - Show reports of working time (reports)  
   + Day report  
   + Week report  
 494 Zeichen (maximal 4000)

**Recent Changes (English)**  
 Versionsname: 0.0.5 alpha  
[\[Weitere Informationen\]](#)  
  
 93 Zeichen (maximal 500)

**Promo Text (English)**   
 0 Zeichen (maximal 80)

**App-Typ**

**Kategorie**

Abb. 11: Informationen zur App (2)

## Geräteverfügbarkeit

### Unterstützte Geräte

<b>Acer</b>	
A100 (vangogh) <a href="#">Ausschließen</a>	Liquid Metal (a4) <a href="#">Ausschließen</a>
E140 (k4) <a href="#">Ausschließen</a>	Picasso (ventura) <a href="#">Ausschließen</a>
E310 (c4) <a href="#">Ausschließen</a>	Picasso (picasso) <a href="#">Ausschließen</a>
<a href="#">Mehr anzeigen...</a>	
<b>Asus</b>	
Eee Pad (EeePad) <a href="#">Ausschließen</a>	Slider SL101 (SL101) <a href="#">Ausschließen</a>
Garmin-Asus A10 (a10) <a href="#">Ausschließen</a>	Transformer TF101 (TF101) <a href="#">Ausschließen</a>
Garmin-Asus A50 (a50) <a href="#">Ausschließen</a>	Transformer TF101G (TF101G) <a href="#">Ausschließen</a>
<a href="#">Mehr anzeigen...</a>	
<b>Casio</b>	
Casio G'zOne Commando (C771) <a href="#">Ausschließen</a>	
<a href="#">Mehr anzeigen...</a>	
<b>Compal</b>	
Andy (cap2) <a href="#">Ausschließen</a>	
<a href="#">Mehr anzeigen...</a>	
<b>Coolpad</b>	
8810 (dkb) <a href="#">Ausschließen</a>	

### Manuell ausgeschlossene Geräte

Abb. 12: Liste der Unterstützten Geräte (mit Möglichkeit manuell bestimmte Geräte auszuschließen)

## Tag 5

### Vorgesehene Features / Schritte

- Export als CSV
- Berechtigungen
- Threads
- Dialoge
- Weiterentwicklungsmöglichkeiten

### Threading

Bei zeit- und rechenintensiven Berechnungen oder Aufgaben sollte man die Aufgabe in einen eigenen Thread auslagern. Dadurch bleibt die Anwendung weiterhin durch den Benutzer bedienbar. Man sollte dem Benutzer zumindest auch eine Rückmeldung geben, wie weit die Aufgabe fortgeschritten ist (Fortschrittsanzeige).

Um dies zu realisieren muss eine eigene Klasse entwickelt werden, die sich von der Thread-Klasse ableitet. In der run()-Methode kann dann die auszuführende Aufgabe implementiert werden. Durch einen parametrisierten Konstruktor können die notwendigen Parameter für die Aufgabe übergeben werden. Dazu gehört unter anderem auch ein Handler, mit dem Sie einer Rückmeldung an die Oberfläche weiterreichen können.

### Dialoge

Bestätigungsdialoge können relativ einfach realisiert werden. Google Ressourcen liefert dazu auch ein paar gute Beispiele dazu:

<http://developer.android.com/guide/topics/ui/dialogs.html>

Die Dialoge können zu dem komplett selbst gestaltet werden und eigene Logik enthalten. Android liefert einige Standard-Dialoge mit.

7. ProgressDialog (Fortschrittsanzeige)
8. DatePicker Dialog (Datumsauswahl)
9. TimePicker Dialog (Zeitauswahl)
10. AlertDialog (ein konfigurierbare Dialog mit 1-3 Buttons).

```
AlertDialog.Builder confirmDialog = new AlertDialog.Builder(this);
confirmDialog.setMessage(getString(R.string.dlg_msg_confirm_delete))
    .setCancelable(true)
    .setIcon(R.drawable.dlg_alert)
    .setTitle(R.string.dlg_title_confirm)
    .setPositiveButton(getString(R.string.dlg_btn_yes), new OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            TimeRecordsTable trt = new
TimeRecordsTable(RecordsListActivity.this);
            trt.deleteById(_SelectedItemId);

            Init();
        }
    })
    .setNegativeButton(getString(R.string.dlg_btn_no), new OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
AlertDialog confirm = confirmDialog.create();
confirm.show();
```

## Entwicklungstipps

- Die erste Version einer App nicht zu komplex gestalten. Die erste Version sollte im besten Fall in 3-4 Wochen realisierbar sein (max. 2 Monate). Dann verliert man nicht die Lust am Programmieren und kann viel schneller auf die Benutzer-Wünsche reagieren.
- Die Weiterentwicklung sollte in etwa in 2-4 Wochen Rhythmus ablaufen. Dadurch bleibt die App lebendig (die Bugs sollten natürlich schneller korrigiert werden :-)).
- Man sollte so früh wie möglich sich für ein Versionskontrollsystem entscheiden (GIT, Mercurial, Subversion, ...).
- Eventuell sollte man sich auch für einen Host für das bevorzugte Versionskontrollsystem entscheiden, um den Code auch von anderen Rechnern erreichen zu können (GitHub, GoogleCode, ...). Eventuell kann auch eigener NAS-Server diese Aufgabe übernehmen (für Synology und QNap Geräte gibt es Erweiterungen für die meisten Versionskontrollsysteme).
- Die Vermarktung der App sollte über mehrere Wege führen:
  - Android Market
  - Androidpit (in deutschen Sprachraum sehr erfolgreich)
  - Eigene Homepage
  - Amazon AppStore
  - ...
- Geld Verdienen mit den Android-App kann über folgende Wege erfolgen:
  - Eine kostenlose Light-Version (um neue Kunden zu werben) und eine Pro-Version mit mehr Funktionen (Funktionsabstand zwischen Light und Pro sollte 2-5 Versionen liegen).
  - Eine kostenlose Basisversion zu der Erweiterungen kostenpflichtig angeboten werden:
    - z.B.: Schnittstellen zu unterschiedlichen Webservice als kostenpflichtige Erweiterung.
  - Eine kostenlose werbefinanzierte Version (z.B. über AdMob) und eine kostenpflichtige werbefreie Version.
  - Eine App mit Spendemöglichkeit (z.B. Flattr, PayPal usw.).
- Die App sollte nicht unbedingt mit Features den Benutzer erschlagen, sondern lieber den Benutzer mit freundlicher Bedienung und einem schlüssigen Konzept überzeugen (App, Erweiterungen, Widgets usw.).
- Setz die App für möglichst viele Geräte um. Eine App, die auch LowBudget Geräte berücksichtigt kann genau den Mehrwert haben, um Ihre App von der Konkurrenz abzuheben.

## Vorschläge zur Weiterentwicklung

- Widgets
  - Erfassung über ein Widget starten
- Typ der Erfassung bestimme (Urlaub, Überstungen, Feiertage usw.)
  - Typ bei Report-Erstellung berücksichtigen
- Einstellungen der App

- Zeichengröße
  - Theme
  - Pausenlänge
  - Wochen/Tagesarbeitsstunden
- Export
  - Export zu CSV, XML usw. auf SD-Karte vom Reports, Erfassungsdaten, komplette DB
  - Versand per Email
- Import
  - Import der Erfassungsdaten (projektbezogen)
  - Import der Feiertage (alle Projekte/projektbezogen)
- Benutzerverwaltung für Tablets
- Projektvorauswahl beim Start über Geo-Lokalisierung
- Multiselect und Funktionen dafür
  - Löschen
  - Setzen bestimmter Markierungen

# Performance Optimierungen

- Zugriff auf Klassenvariablen (Eigenschaften) direkt über die Klassenvariablen statt über die Getter und Setter. Performance-Steigerung liegt ca. beim Faktor 2-3.
- Bei Zusammenfügen von String-Ketten mit Hilfe des StringBulders, statt des „+“-Operators. Mit der Anzahl der Konkatenationen steigt der Performance-Verlust beim „+“-Operator exponentiell an, da jedes Mal ein neues Objekt erzeugt werden muss. Beim StringBuilder steigt die Rechenzeit dagegen nur linear an.
- Es sollen unnötige Objekterzeugungen vermieden werden. So z.B.:
  - Objekterzeugung in den Schleifen
  - Mehrfache Objekterzeugung von oft benutzen Objekten in der Klasse. Diese sollten am besten direkt als Klassenvariablen, und somit nur einmal, initialisiert werden.
- Schleifen optimieren. Statt:  
**for (int i = 0; i < array.length; i++) { ... }**  
sollte  
**int arrayLenght = array.length; for (int i = 0; i < arrayLength; i++) { ... }**  
verwendet werden.
- Layoutoptimierung mit dem mitgelieferten Tool `layoutopt` durchführen.
- Größere (mehr als 100 Elemente) Arrays sind deutlich performanter als ArrayList Objekte.
- Einfache Klassen, sollten wenn möglich durch Arrays ersetzt werden, da dadurch die Objekterzeugung entfällt.
- Wenn möglich, sollten final und static Zugriffsmodifizierer verwendet werden.
- Die Schleifen, nach dem das gewünschte Ergebnis erreicht wurde, sollten durch `break` oder `continue` verlassen werden und nicht unnötig die restlichen Elemente durchlaufen.
- Rechenintensive oder Zeitintensive Aufgaben sollten immer in ein eigenes Thread ausgelagert werden, damit der Benutzer weiterhin die Kontrolle über die App behält.
- Beim WebServices und Netzwerkzugriffen sollte dem kompakteren JSON-Format immer der Vorzug vor dem XML-Format gewährt werden (kleinerer Overhead).
- Benutzung des mitgelieferten Obfuscators ProGuard, um den kompilierten Code zu optimieren (Komprimierung von Code, weglassen unbenutzer Methoden / Eigenschaften / Klassen, kürzung der Bezeichner usw.).
- Test der eigenen Algorithmen auf Performance und Test der Alternativen (Benchmarking).
- Benutzung von TraceView (DDMS), um Flaschenhalse beim Speicher und Ausführungszeit zu finden.
- Bei den SQL-Abfragen sollten nur die wirklich notwendige Daten abgefragt werden (nur benötigte Spalten und nur die benötigte Datenmenge, Zeilenanzahl).



# Anhang

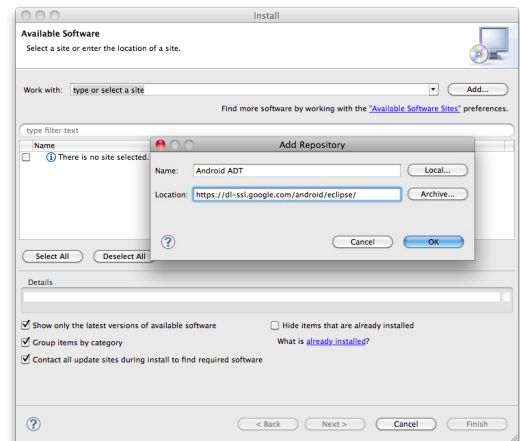
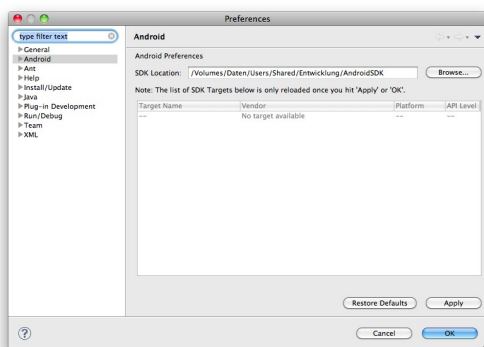
## Installation von Eclipse IDE und ADT

### Installation von Eclipse

- Eclipse Classic ab Version 3.5 herunterladen (<http://www.eclipse.org/downloads/>).
- Die heruntergeladene Datei in einen Ordner entpacken (z.B.: C:\Entwicklung\Eclipse unter Windows, oder /Users/Shared/Entwicklung/Eclipse unter Mac OS X).
- Eclipse starten und das Standard-Workplace (z.B.: C:\Entwicklung\Android unter Windows oder /Users/Shared/Entwicklung/Android unter Mac OS X) vergeben.

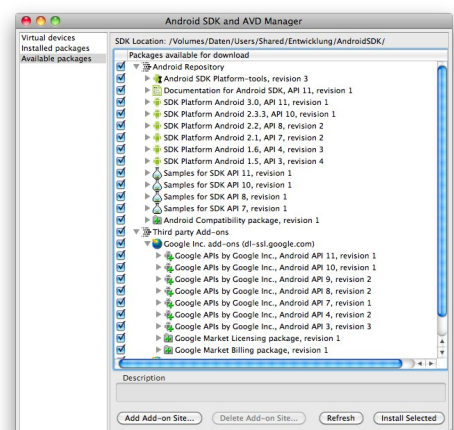
### Installation von ADT

11. Android SDK für das gewünschte Betriebssystem herunterladen (<http://developer.android.com/sdk/index.html>).
12. Die heruntergeladene Datei in einen Ordner entpacken (z.B.: C:\Entwicklung\AndroidSDK unter Windows oder /Users/Shared/Entwicklung/AndroidSDK unter Mac OS X).
13. Eclipse starten und zu **Help** → **Install New Software** gehen.

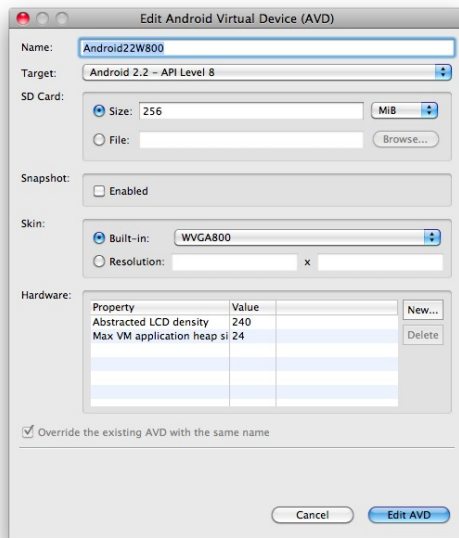


14. Bei „**Work with:**“ auf „**Add...**“ klicken, um neues Software-Repository hinzuzufügen (siehe Bild).
15. Namen frei vergeben und als „Location“ <https://dl-ssl.google.com/android/eclipse/> eingeben
16. **OK** klicken.
17. Developer Tools als zu installierende Komponenten auswählen (es dauert einige Zeit, bis die Repository eingelesen wird).

18. Die Installation mit „**Next**“ → „**Next**“ → „**Accept**“ und „**Finish**“ abschließen.
19. Eclipse neu starten.
20. Gehen Sie zu Preferences (unter Windows über **Window** → **Preferences**, unter Mac OS X über **Eclipse** → **Preferences** zu erreichen).
21. Gehen Sie zum Punkt „Android“ und tragen Sie da



den Speicherort von Android-SDK ein (nach dem Vorschlag wäre es unter Windows der Ordner `C:\Entwicklung\AndroidSDK` und unter Mac OS X `/Users/Shared/Entwicklung/AndroidSDK`).



22. Nun gehen Sie zu „**Window**“ → „**Android-SDK and AVD Manager**“ → „**Available packages**“ und installieren Sie die benötigten (z.B.: Android 1.6 und Android 2.2) oder alle SDK-Versionen.
23. Nach dem Neustart von Eclipse gehen Sie wieder zu „**Android-SDK and AVD Manager**“.
24. Legen Sie unter „**Virtual Devices**“ die virtuellen Android-Geräte, für die Sie entwickeln möchten (z.B.: ein Android 2.2 Gerät mit WXGA Auflösung [800x480] und 2GB SD-Karte). Sie können auch mehrere virtuelle Geräte anlegen und beim Start eines Android-Projektes dann eins davon auswählen.
25. Damit sind alle Vorbereitungen für die Android-Programmierung getroffen.

## Git – Versionsverwaltung unter Eclipse

Quelle: <http://de.wikipedia.org/wiki/Git>

Git (dt. [ɡɪt]) ist eine Freie Software zur verteilten Versionsverwaltung von Dateien, die ursprünglich für die Quellcode-Verwaltung des Linux Betriebssystemkerns entwickelt wurde.

Die Entwicklung von Git wurde im April 2005 von Linus Torvalds begonnen, um das bis dahin verwendete Versionskontrollsystem BitKeeper zu ersetzen, das durch eine Lizenzänderung vielen Entwicklern den Zugang verwehrte. Die erste Version erschien bereits wenige Tage nach der Ankündigung.

Git läuft auf fast allen modernen UNIX-artigen Systemen, wie Linux, Solaris, Mac OS X, FreeBSD, DragonFly BSD, NetBSD, OpenBSD, AIX und IRIX. Auf Microsoft Windows läuft es entweder mit Hilfe der Cygwin-Umgebung oder mit Msysgit, bzw. der seit einiger Zeit verfügbaren TortoiseGit-Shell-Erweiterung (ähnlich TortoiseSVN).

Die derzeit aktuelle Version wird produktiv für die Entwicklung des Linux-Betriebssystemkerns und im Rahmen vieler weiterer Projekte eingesetzt, darunter Git selbst, Samba, X.Org, Qt, GNOME, One Laptop per Child, Ruby on Rails, VLC, Wine, LilyPond, TaskJuggler, DragonFly BSD, Android, BusyBox, Amarok, Perl 5, x264 und phpBB. Derzeitiger Maintainer von Git ist Junio Hamano.

### Installation des Plug-ins

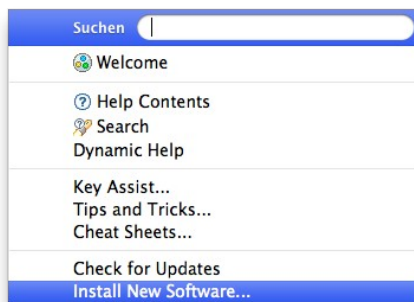


Abb. 13: Install New Software

Klicken Sie unter „Help“ auf den Punkt „Install New Software“ um zum Installationsdialog für Erweiterungen von Eclipse zu gelangen (Abb. 1).

Unter „Work with“ (Abb. 2) wählen Sie „--All Available Sites--“ um alle verfügbaren Quellen für Eclipse einzu-

beziehen. Die Aktualisierung der Liste kann abhängig von Rechenleistung und Internetverbindung mehrere Minuten dauern. Geben Sie in der Suchmaske „git“ ein. In der Ergebnisliste markieren Sie die Plug-ins JGit (Java-Implementierung des Git-Clients) und EGit (Team-Plug-in für Eclipse). Klicken Sie auf „Next“.

Nach der Überprüfung der Abhängigkeiten erscheint ein Bestätigungsdialog mit der Auflistung aller zu installierenden Komponenten (Abb. 3). Klicken Sie hier wieder auf „Next“.

Im nächsten Dialog (Abb. 4) bestätigen Sie die Lizenzvereinbarung mit „I accept the terms of the license agreement“ und dem Klick auf „Finish“.

Nach der Installation (Abb. 5) muss Eclipse neu gestartet werden (Abb. 6).

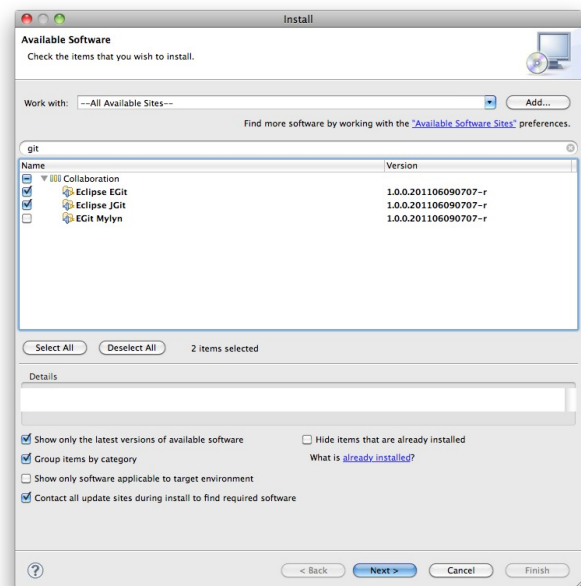


Abb. 14: Auswahl der Plug-ins

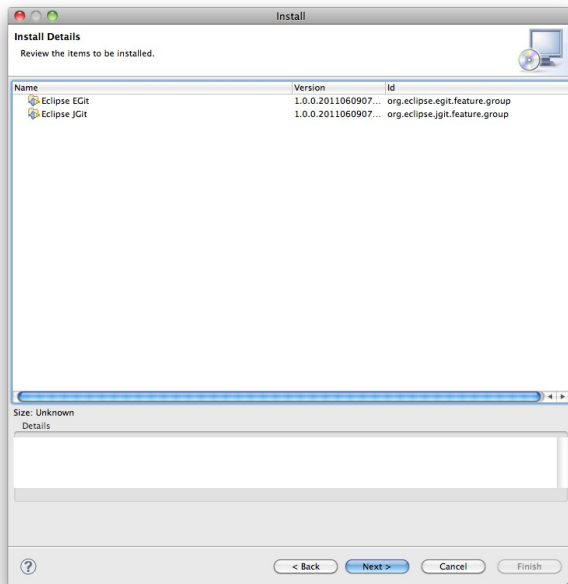


Abb. 15: Zu installierende Komponenten

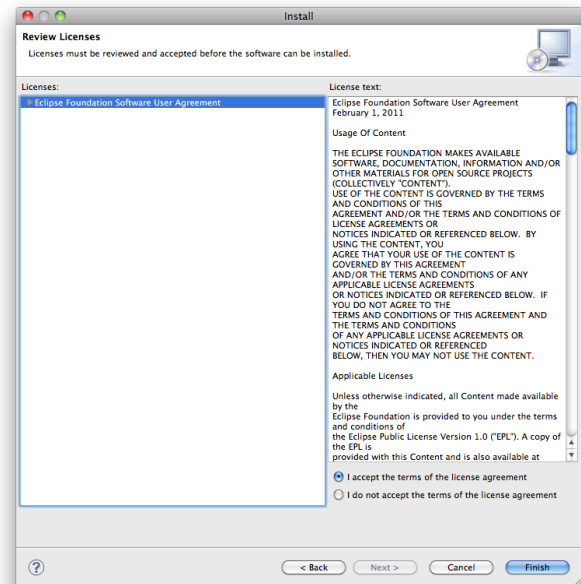


Abb. 16: Lizenzbestätigung

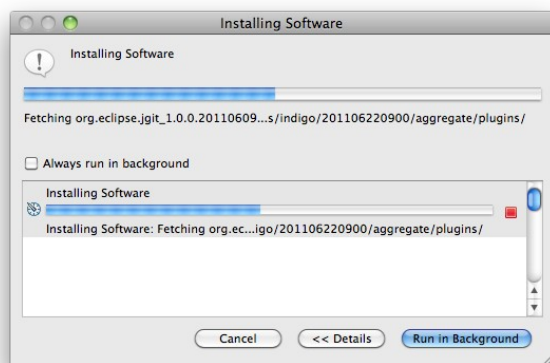
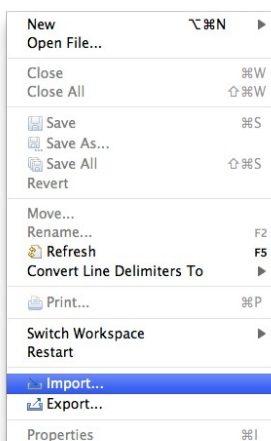


Abb. 17: Installation



Abb. 18: Neustart der Entwicklungsumgebung

## Import eines Projektes



Zum Import eines Git-Projektes (lokal oder hier von GitHub) klicken Sie zuerst unter „File“ auf den Menüpunkt „Import“ (Abb. 7).

Als Quelle wählen Sie in der Liste unter „Git“ den Punkt „Projects from Git“ und klicken Sie auf „Next“ (Abb. 8).

Im nächsten Dialog (Abb. 9) klicken Sie auf „Clone“

um die Adresse des Git-Repositories anzugeben (Abb. 10). Die notwendige Adresse bei Github unter der Projektadresse (z.B. wie in Abb. 11 dargestellt).

Klicken Sie auf „Next“.

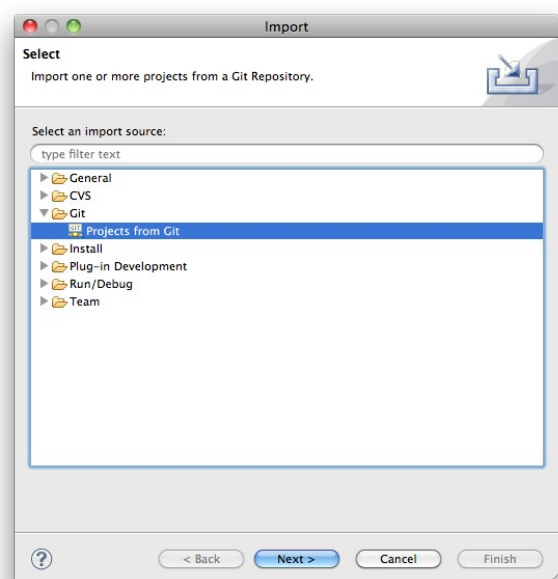


Abb. 20: Auswahl der Quelle

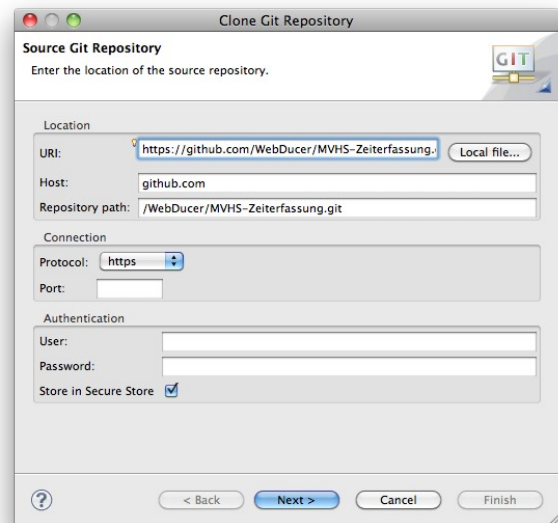
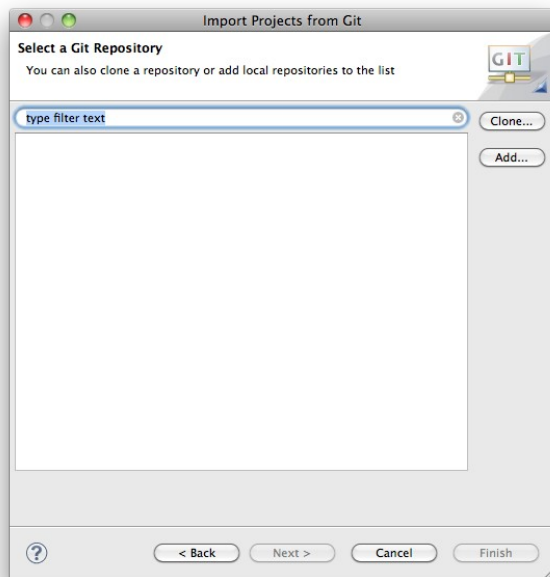


Abb. 22: Adresse der Git-Repository

Abb. 21: Clone-Adresse auswählen

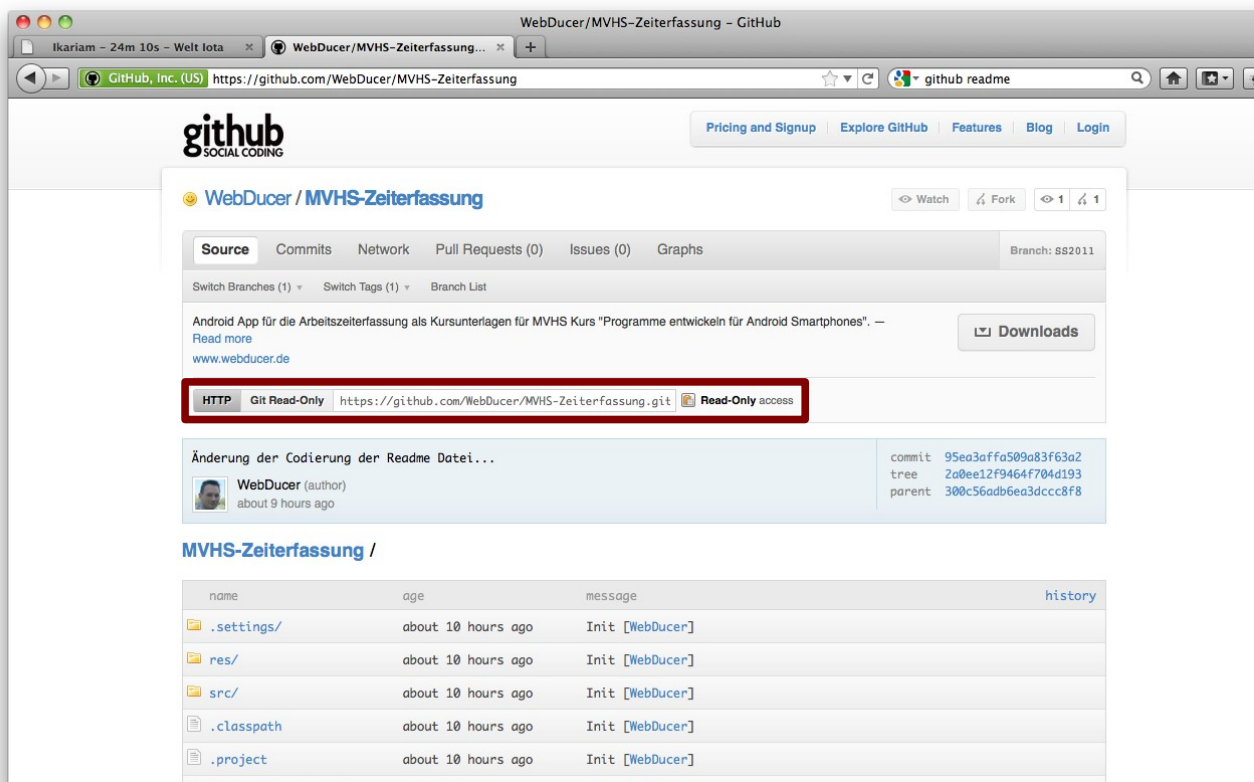


Abb. 23: GitHub - Repository-Adresse

Im nächsten Dialogfenster (Abb. 12) können Sie aus den verfügbaren Branches (Abzweigungen des Quellcodes) da zu importierende auswählen. Klicken Sie danach wieder auf „Next“.

Das folgende Dialog (Abb. 13) bietet die Möglichkeit den Speicherort des Projektes auszuwählen. Weiterhin ist hier auch möglich einzustellen, welcher Branch nach dem Import als das Aktuelle markiert wird (wenn mehrere Branches importiert werden) und wie das Quell-Repository bezeichnet werden soll. Klicken Sie, nach dem Sie alle Angaben Ihre Wünschen angepasst haben, auf „Next“.

Im nächsten Dialog haben Sie die Möglichkeit, falls erwünscht, eine Verknüpfung mit dem Code-Review Service „Gerrit“ herzustellen. Klicken Sie hier anschließend auf „Finish“.

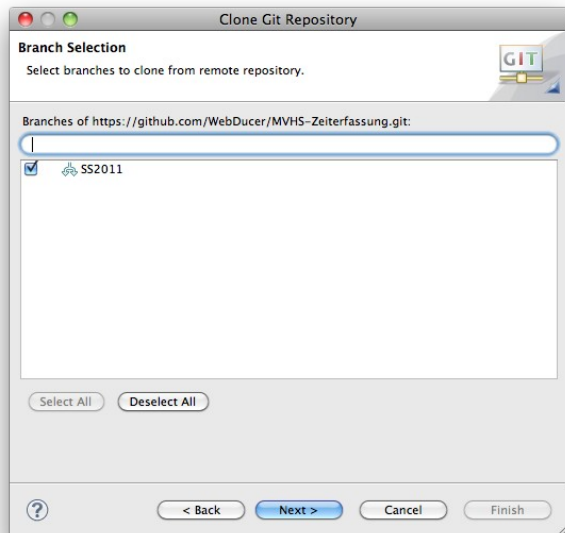


Abb. 24: Auswahl der Abzweigungen

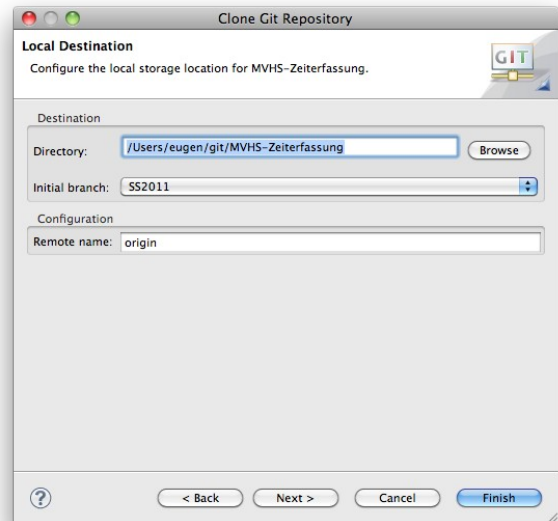


Abb. 25: Speicherort des Projektes

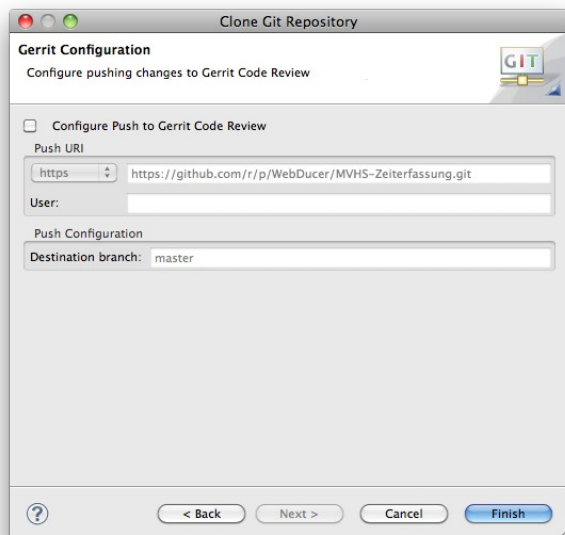
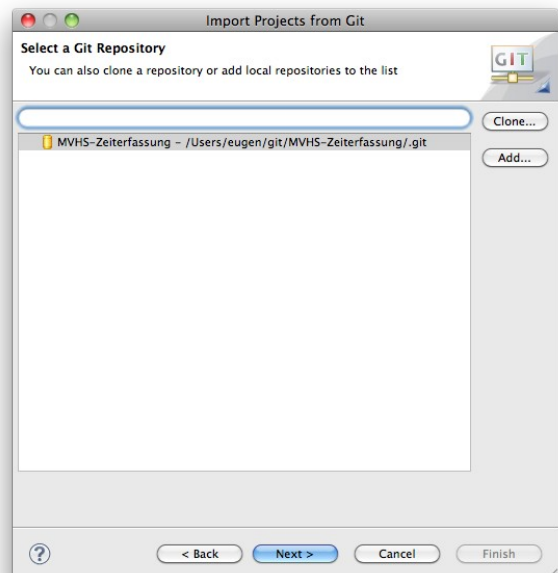


Abb. 27: Gerrit Codereview



In der Liste erscheint nur das gerade angelegte Git-Repository (Abb. 15). Wählen Sie dieses aus und klicken Sie auf „Next“.

Im nächsten Dialog (Abb. 16) sehen Sie das Arbeitsverzeichnis, das aus dem Git-Repository importiert wird. Klicken Sie hier wieder auf „Next“.

Im letzten Schritt können Sie auswählen, welche Projekte aus dem Arbeitsverzeichnis importiert werden sollten (Abb. 17). Klicken Sie auf „Finish“.

Das Projekt steht nun in Workspace zur Verfügung.



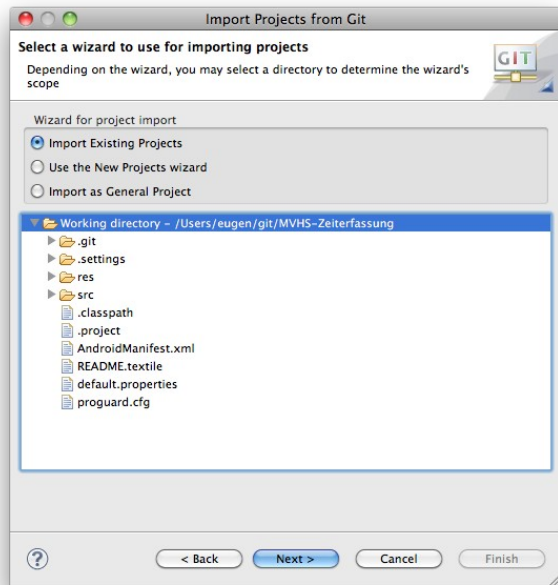


Abb. 28: Importverhalten

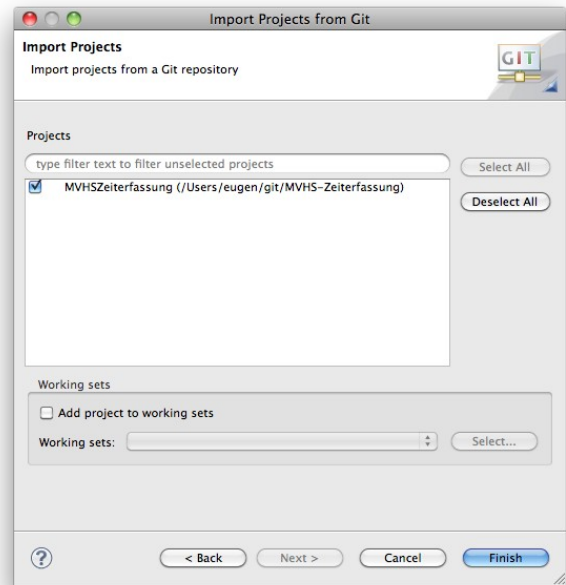


Abb. 29: Projekte auswählen