

4. Regionalwettbewerb „Jugend forscht“ in Jena, Fachgebiet
Mathematik/Informatik

**Wettkampferfassung und -auswertung
beim Kanusport mittels eines kostenlos
und einfach einrichtbaren
Softwaresystems**

Eric Ackermann
Carl-Zeiss-Gymnasium Jena

15. Januar 2017

betreut durch:
Herrn Bernd Schade

Kurzfassung

Professionelle Zeitmesssysteme für den Kanusport sind teuer und meist kompliziert einzurichten. Kleinere Vereine, die sich ein solches System nicht leisten können, sind auf Papierlisten für Strafzeiten und die Zeitmessung per Hand angewiesen. Dies bedeutet einen hohen Zeit- und Organisationsaufwand bei Training und Wettbewerben.

Ziel dieses Projekts ist die Entwicklung eines Softwaresystems, welches auf vorhandener bzw. kostengünstig anzuschaffender Hardware funktioniert, einfach zu bedienen ist und die Zeitmessung für Kanusportvereine optimiert. Im Mittelpunkt steht die Entwicklung einer Android-App, die das Eintragen von Strafzeiten sowie Start- und Zielzeit für die einzelnen Startnummern und Tore mit einem Klick leistet. Die in der App erhobenen Daten werden über ein lokales WLAN-Netzwerk unter Zuhilfenahme eines lokalen PHP- und MySQL- kompatiblen Servers mit einem zentralen Laptop synchronisiert. Auf diesem läuft eine eigens entwickelte Desktopanwendung, die alle Daten zusammenführt, jederzeit anzeigt und nach dem Wettkampf mit einem Klick übersichtlich auswertet.

Abkürzungsverzeichnis

PHP Hypertext PreProcessor

SQL Structured Query Language

IP Internet-Protokoll

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

POI Poor Obfuscation Implementation

Inhaltsverzeichnis

1	Einleitung	1
1.1	Zeitmessung im Kanusport/Problemstellung	1
1.2	Konzeption	2
2	Vorgehensweise	3
2.1	Grundprinzip der Vernetzung der verschiedenen Geräte	3
2.2	Aufbau und Funktionsweise der verwendeten MySQL-Datenbank	4
2.3	Das Hauptprogramm	5
2.3.1	Features	6
2.3.2	Anlegen und Verwalten der Datenbank	6
2.3.3	Datenverarbeitung und Auswertung	7
2.4	Die Android-App	7
2.4.1	Funktionen	8
2.4.2	Umsetzung des Verbindungsaufbaus	8
2.5	Der PHP-Server	9
2.5.1	Aufgabe	9
2.5.2	Umsetzung des Zugriffes auf die Datenbank	9
2.6	Technische Umsetzung der Synchronisation	10
2.7	Beschreibung des Ablaufs einer Verwendung des Softwaresystems	11
2.8	Aufgetretene Probleme und ihre Lösungen	12
3	Zusammenfassung und Schluss	14
3.1	Umsetzung der Konzeption	14
3.2	Ausblick	15
	Glossar	16
	Literatur	17
4	Anhang	18
4.1	Funktionsweise der Hauptmethode, welche alle Daten zusammenführt	18
4.2	Protokollierung und Wiederherstellungsmodus	18
4.3	Screenshots	18
4.3.1	Hauptprogramm	18
4.3.2	App	21
4.3.3	Protokolle	23
4.4	Sourcecode	23

Abbildungsverzeichnis

1.1	Zeitmesssystem „Basic“ von IMAS, vgl. [3]	1
2.1	Schema der Vernetzung der verschiedenen Geräte, vgl. [2, 9, 7, 1, (Illustrationen)]	3
2.2	Übersicht über die Datenbank „android_connect“ und ihre Tabellen in einer Beispielbelegung, Darstellung durch verwendetes Datenbankverwaltungsprogramm PHPMyAdmin (Teil von XAMPP)	4
2.3	Beispielbelegung der Datenbanktabelle „allgemein“	5
2.4	Beispielbelegung der Datenbanktabelle „namen“	5
2.5	Beispielbelegung der Tabelle eines Messtores	5
2.6	Ausschnitt aus einer Auswertungsdatei mit Beispieldaten, dargestellt von LibreOffice	7
4.1	Konfigurationsfenster des Hauptprogramms mit Beispieleingaben	18
4.2	Wahl der Kategorie, welche starten soll	19
4.3	Zuordnung der Messtore zu Messstation 1	19
4.4	Hauptfenster vor Start des ersten Starters	19
4.5	Hauptfenster während eines Laufs	20
4.6	Hauptfenster bei Laufende	20
4.7	App vor dem Verbindungsaufbau	21
4.8	App bei der Funktionswahl	22
4.9	App im Startmodus bei Auswahl der Startnummer (Stopppmodus ähnlich)	22
4.10	App beim Eintragen einer Strafzeit	23
4.11	Protokoll eines Programmlaufs mit Beispielausgaben	23

Tabellenverzeichnis

2.1	Features des Hauptprogramms „Android_Connector“	6
2.2	Features der App „Android Connector App HTTP“	8

1 Einleitung

1.1 Zeitmessung im Kanusport/Problemstellung

- Notwendigkeit der Messung und Zusammenführung von Zeiten beim Kanuslalom: Laufzeiten von Start zu Ziel sowie Strafzeiten beim falschen Passieren von Toren erheben, zuordnen zu Starter, Kategorie und Lauf, vergleichen...
- bei jedem Wettkampf und im Training auch bei kleinen Vereinen nötig, so auch beim Jenaer Kanu- und Ruderverein und beim Kanu-Slalom-Teil des SV Schott
- dort: Erhebung der Strafen an 18-25 Messtoren einer ca. 200 m langen Strecke durch Helfer an bis 8 Messstationen bei 27 Rennen (festgelegt durch z.B. Bootstyp und Alter der Sportler) mit ggf. dutzenden Startern → hoher Organisationsaufwand in Vorbereitung und Wettkampfauswertung
- bisher: Aufschreiben von Strafen per Hand, Auswertung über Excel-Makros
- → Zeitaufwand, Organisationsaufwand, Fehleranfälligkeit
- Existenz von kommerziellen Systemen zur automatischen Zeiterfassung
- vollwertige Anlage z.B. mit „Time System Professional“ von IMAS[4, 5] → Zeiterfassung per Kamera (Ziel, siehe Abbildung 1.1) und optional automatischer Startanlage bzw. Startkoffer (500€), Erfassung von Zwischenzeiten (nicht nötig), Verwaltung der Starter per Software, Livestream der Werte ins Internet, Vernetzung der Geräte per LAN oder WLAN



Abbildung 1.1: Zeitmesssystem „Basic“ von IMAS, vgl. [3]

- aber: Fehlen der Synchronisation mit Strafzeiten, keine Erfassung dieser
- bis zu 6 Laptops benötigt mit jew. 1 Bediener
- IMAS: Preis auf Anfrage, da optionale kostenpflichtige Zusatzgeräte wie eine Liveanzeige am Wettbewerbort
- Vergleichbares System von Lynx kostet aber bis zu 41.595,- € zzgl. MwSt, in der günstigsten Variante noch 11.995,- € zzgl. MwSt[8]; ebenfalls keine Synchronisation von Strafzeiten

- Zusammenfassung: kein kostengünstiges Angebot für kleine Vereine gefunden
- kommerzielle Systeme bieten auf der einen Seite meist mehr Funktionen als grundlegend nötig, daher teuer
- auf der anderen Seite: lassen wichtige Features (Strafzeiten) vermissen
- Grund: Systeme für Kanu-Sprint gebaut, bei dem Starter alle gleichzeitig starten und eine Strecke möglichst schnell entlangfahren müssen, die gerade ist
- in Jena: Kanuslalom -> Starter starten nacheinander, müssen Strecke mit Toren präzise folgen -> etwas andere Anforderung
- Frage der Arbeit: Ist das Problem mit kostenloser, selbst entwickelter Software und günstiger Hardware zu lösen?

1.2 Konzeption

- Problem der Starterverwaltung, Wettkampfzeiterhebung, Strafzeitensynchronisation und Wettkampfauswertung möglichst günstig und einfach zu lösen
- dafür: Verwendung von Android-Smartphones (fast jeder besitzt eines), einem Windows-Rechner (auch sehr wahrscheinlich vorhanden), gratis-Software
- Vernetzung der Geräte mit einem drahtlosen Netzwerk

2 Vorgehensweise

2.1 Grundprinzip der Vernetzung der verschiedenen Geräte

- zentraler Laptop: wichtigstes Gerät im entwickelten Softwaresystem
- Windows-Gerät; nötige Software auch auf älteren Geräten bis zu Win XP lauffähig, Kompatibilität zu Linux- oder Apple-Geräten im Rahmen der grundlegenden Funktionalität gegeben (ggf. Verlust von Zusatzfunktionen)
- Laptop baut WLAN für Vernetzung der Geräte auf (wenn er genug Reichweite hat, sonst Repeater), aber: kein Internetzugriff benötigt

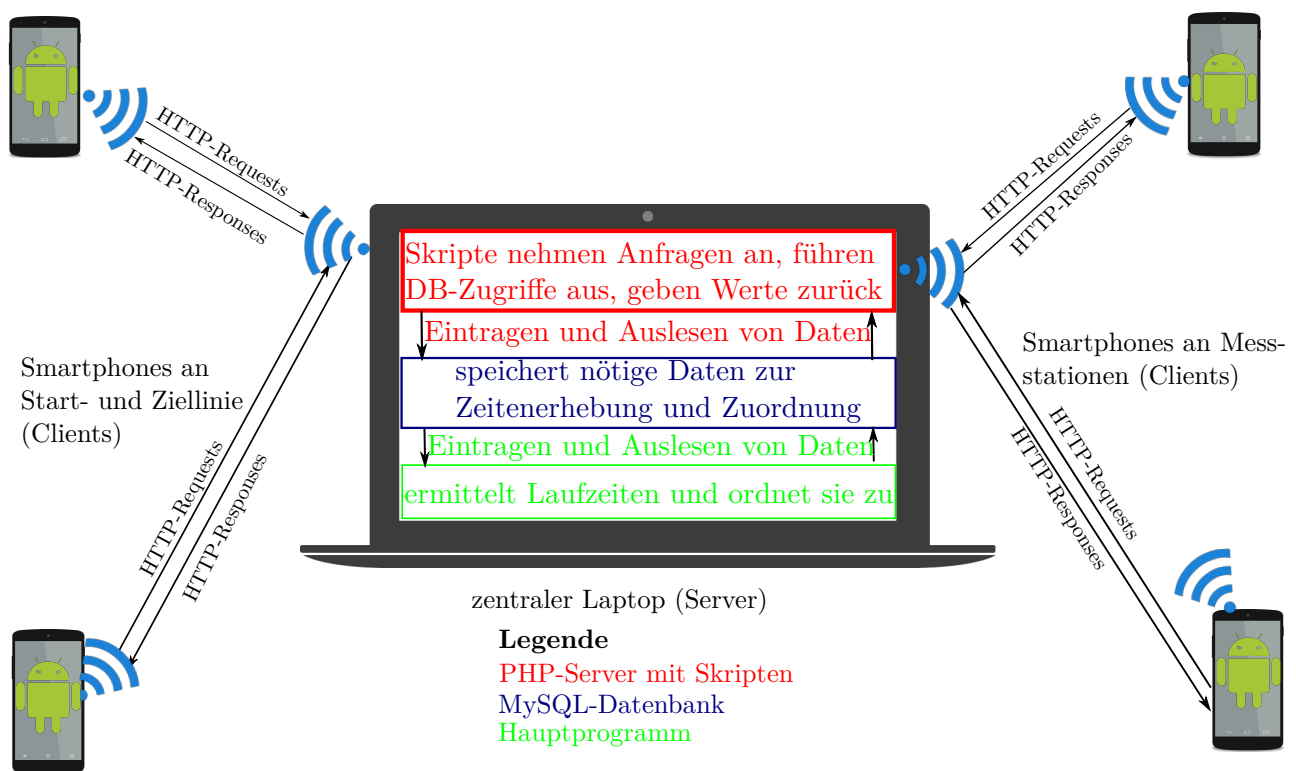


Abbildung 2.1: Schema der Vernetzung der verschiedenen Geräte, vgl. [2, 9, 7, 1, (Illustrationen)]

- zentral im Projekt steht MySQL-DB, die alle Daten speichert (siehe Abbildung 2.1)
- Datenbank wird durch MariaDB verwaltet (freier Ableger von MySQL mit gleicher technischen Grundlage)
- parallel zu DB läuft ein PHP-Server, hier der Apache Webserver
- Helfer mit Smartphones in ihren Messstationen, auf Smartphones wird Android-App ausgeführt, die Start-Stopp oder Eintragen von Strafzeiten ermöglicht
- App kommuniziert über den PHP-Server mit der Datenbank

- auf dem Hauptrechner läuft in JavaFX geschriebenes Hauptprogramm
- verbindet sich direkt mit DB, kann über sie mit App synchronisieren
- verwaltet Starter und Rennen, ermittelt während eines Laufs immer die aktuellen Laufzeiten der laufenden Starter, wertet am Ende den Wettkampf aus

2.2 Aufbau und Funktionsweise der verwendeten MySQL-Datenbank

- MariaDB kann nur auf Webserver ausgeführt werden → zentraler Laptop, der hier auch als Server fungiert, wird zum Anlegen eines Webserver genutzt
- dafür: Installation der Software XAMPP → legt Webserver an, der Verwalten einer Datenbank und Ausführung von PHP ermöglicht
- Webserver bleibt rein lokal, kann innerhalb des lokalen Netzes adressiert werden, aber nicht vom Internet aus (Datensicherheit), solange kein Gerät eine Internet-Verbindung besitzt
- direkte Internet-Zugriffsmöglichkeit nicht zu empfehlen, da XAMPP für einfachen und schnellen lokalen Betrieb konzipiert ist und nicht für den Betrieb als echter Server → evtl. Sicherheitslücken
- DB (siehe Abbildung 2.2) enthält genannte Tabellen: „namen“, „allgemein“, für jede Messstation 1, sowie eine für die Auswertung (nicht im Bild)



Tabelle	Aktion	Datensätze	Typ	Kollation	Größe
allgemein	Anzeigen Struktur Suche Einfügen Leeren Löschen	7	InnoDB	latin1_swedish_ci	16 KiB
messstation_0	Anzeigen Struktur Suche Einfügen Leeren Löschen	2	InnoDB	latin1_swedish_ci	16 KiB
messstation_1	Anzeigen Struktur Suche Einfügen Leeren Löschen	1	InnoDB	latin1_swedish_ci	16 KiB
messstation_2	Anzeigen Struktur Suche Einfügen Leeren Löschen	0	InnoDB	latin1_swedish_ci	16 KiB
messstation_3	Anzeigen Struktur Suche Einfügen Leeren Löschen	2	InnoDB	latin1_swedish_ci	16 KiB
messstation_4	Anzeigen Struktur Suche Einfügen Leeren Löschen	0	InnoDB	latin1_swedish_ci	16 KiB
messstation_5	Anzeigen Struktur Suche Einfügen Leeren Löschen	2	InnoDB	latin1_swedish_ci	16 KiB
messstation_6	Anzeigen Struktur Suche Einfügen Leeren Löschen	1	InnoDB	latin1_swedish_ci	16 KiB
messstation_7	Anzeigen Struktur Suche Einfügen Leeren Löschen	2	InnoDB	latin1_swedish_ci	16 KiB
namen	Anzeigen Struktur Suche Einfügen Leeren Löschen	2	InnoDB	latin1_swedish_ci	16 KiB
10 Tabellen	Gesamt	19	InnoDB	latin1_swedish_ci	160 KiB

Abbildung 2.2: Übersicht über die Datenbank „android_connect“ und ihre Tabellen in einer Beispielbelegung, Darstellung durch verwendetes Datenbankverwaltungsprogramm PHPMyAdmin (Teil von XAMPP)

- Tabelle „allgemein“: speichert Informationen, die für die Initialisierung und korrekte Eintragung der Handy-App relevant sind
- namentlich (siehe Abbildung 2.3): ob der erste Starter bereits gestartet ist → ab dann Eintragung von Strafen möglich in App, Nummer des aktuellen Laufs (wird zur Orientierung in der App eingeblendet), die Tore, welche zu den einzelnen Messstationen gehören, die Startzeit des Wettkampfes als Java-Timestamp und die Anzahl der Messstationen, damit Benutzer der App sich an jede davon anmelden können

	Attribut	Wert
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	gestartet	true
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	lauf	0
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Messstation_0_Tore	0 3 5 7
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Messstation_1_Tore	2 4
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Messstation_2_Tore	1 6
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Startzeit	1482847016286
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Zahl_Stationen	3

Abbildung 2.3: Beispielbelegung der Datenbanktabelle „allgemein“

- Tabelle „namen“ (siehe Abbildung 2.4): speichert Namen und Startnummer der Starter für Anzeige oder Auswahl der Starter in der App, eine ggf. von der App per Script eingetragene Start- oder Zielzeit als PHP-Timestamp sowie eine Zusammenfassung der einzelnen Läufe für eine spätere Auswertung als String, zudem: Information, welchen Lauf der Wiederholungslauf ersetzt, enthalten

	Namen	Startnummer	Startzeit	Zielzeit	Lauf_1	Lauf_2	Lauf_Wiederholung	Wiederholung_ersetzt
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Hans-Dietrich Genscher	10	1482847027.7587	1482847166.0294	2:18,193 104 4:2,193 Tor: 1 Strafe: 50 Tor: 2 Straf...	NULL	NULL	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	Franz Franz	27	1482847020.1146	1482847159.4347	2:19,320 104 4:3,320 Tor: 1 Strafe: 0 Tor: 2 Straf...	NULL	NULL	NULL

Abbildung 2.4: Beispielbelegung der Datenbanktabelle „namen“

- Tabellen für jew. Messtore (Nummer in Computerzählweise ab 0, siehe Abbildung): speichert an diesem Tor je die Startnummer und ihre Strafzeit

	Startnummer	Zeitpunkt
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	10	50
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	27	0

Abbildung 2.5: Beispielbelegung der Tabelle eines Messtores

- Tabelle „auswertung“ wie Tabelle „namen“, nur dass sie alle Startnummern enthält und bei allen die Kategorie mit angegeben ist; keine Spalten für Start- und Zielzeit
- Startnummer in allen genannten Tabellen der Primärschlüssel

2.3 Das Hauptprogramm

- Programm entwickelt mit NetBeans IDE 8.2, läuft in JVM → prinzipiell zu allen Betriebssystemen kompatibel, Ausführung unter Windows empfohlen
- läuft auf Hauptrechner, 1 Person zur Bedienung ausreichend, muss nur je nächsten Lauf/nächste Kategorie starten

- über WLAN indirekt mit App verbunden

2.3.1 Features

- Tabelle 2.1 zeigt alle Features des Hauptprogramms zusammengefasst; bei Interesse an Umsetzung steht Quellcode zur Verfügung, entsprechende Klassen und Methoden sind zugeordnet

Feature	Realisierung mittels Java-Klasse(n) und Methode(n)
Einlesen von Starterdaten aus Excel-Dateien	Prozedur ConfigWindowController.readExcel(), Klasse ExcelReader, Library Apache POI
Anzeige, Veränderbarkeit eingelesener Daten	Klasse ConfigWindowController, Klasse javafx.scene.control.TableView (und assoziierte Klassen), Klasse Person, Klasse EditableTableCell
automatisches Leeren der Datenbank	Prozedur MySqlConnection.reset
automatisches Starten und Stoppen von XAMPP	Prozeduren ConfigWindowController.xampp_start, xampp_stopp
automatisches Kopieren von nötigen PHP-Skripten, Überschreiben der PHP-Konfiguration	Prozedur ConfigWindowController.overridePHPConfig
Erstellen von Sicherungsprotokollen	Prozedur MySqlConnection.werteFesthalten
Prüfen des Netzwerkzugriffs, Ermittlung der eigenen IP-Adresse	Klasse NetworkUtil
Zugriff auf MySQL-Datenbank (allgemein)	Klasse MySqlConnection, Klasse com.mysql.jdbc.Driver (Teil der verwendeten Library), Klasse java.sql.Statement (für eigentliche Abfragen)
Verwaltung der Starter, Zuordnung zu Kategorien, Start dieser	Klasse ConfigWindowController
Verwaltung der Daten der einzelnen Läufe, Start und Stopp dieser	Klasse FXMLDocumentController
Anzeige der aktuellen Laufaten der jeweiligen Starter	Klasse MySqlConnection, Klasse javafx.scene.control.ListView
Anzeige der Daten im angegebenen Intervall	Klasse java.util.concurrent.ScheduledExecutorService, Klasse java.util.concurrent.ScheduledFuture, Klasse java.lang.Runnable
Auswertung der Rennen	Prozedur ConfigWindowController.printResults, Funktion MySqlConnection.getAuswertung

Tabelle 2.1: Features des Hauptprogramms „Android_Connector“

2.3.2 Anlegen und Verwalten der Datenbank

- genereller Verbindungsaufbau zur Datenbank: im Konstruktor der Klasse MySqlConnection
- Konstruktor speichert übergebene Konfigurationswerte global und legt Instanz der Klasse java.sql.Connection namens conn an, das eine Verbindung zur DB herstellen kann → Verbindung

zur DB ab jetzt möglich, SQL-Abfragen können ausgeführt werden

- 2 Arten von Abfragen: Schreibafragen, die z.B. über Befehl CREATE eine neue Tabelle anlegen oder über UPDATE Werte in einer Tabelle ändern, und Leseabfragen, die ein Ergebnis liefern sollen
- Instanz conn von java.sql.Connection stellt Methode createStatement bereit, das Instanz des Interfaces java.sql.Statement zurückgibt
- Klasse Statement stellt Methode execute bereit, welche einen übergebenen String als Datenbankabfrage in der adressierten Datenbank ausführt (Schreibzugriff)
- zudem: Methode executeQuery, welche übergebene SQL-Abfrage in DB ausführt und Instanz von java.sql.ResultSet zurückgibt
- dieses: repräsentiert von Datenbank erzeugte Antworttabelle, kann Zeilenweise gelesen werden, Zugriff auf einzelne Spalten und Speicherung im String dann auch möglich

2.3.3 Datenverarbeitung und Auswertung

- Ende des Programmlaufs wird dadurch erkannt, dass keine weiteren startbaren Kategorien verbleiben
- dann: zunächst Aufruf der Prozedur printResults im ConfigWindowController, diese ruft Methode getAuswertung von MySQLConnection auf
- liest Laufzusammenfassungen aus Tabelle „auswertung“ der Datenbank aus, ersetzt Läufe ggf. durch den Wiederholungslauf (wie angegeben), gibt Array mit allen Werten aller Starter zurück
- printResults nimmt nun Laufzusammenfassungen auseinander, extrahiert reine Laufzeit, Gesamtstrafen, Gesamtlaufzeit und die Strafen an den einzelnen Toren
- Library Apache POI wird verwendet, um daraus eine Excel-Datei (siehe Abbildung 2.6) zu erzeugen; dabei wird für jede Kategorie eine Mappe angelegt, die Angaben für jeweilige Starter (aus der HashMap gelesen) enthält
- Auswertung nun möglich durch Makro oder z.B. durch Sortierung

A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	Startnummer	Name	Kategorie	reine Laufzeit- Lauf 1	Gesamtstrafen- Lauf 1	Laufzeit insgesamt- Lauf 1	Strafe Tor 1	Strafe Tor 2	Strafe Tor 3	Strafe Tor 4	Strafe Tor 5	Strafe Tor 6	reine Laufzeit- Lauf 2	Gesamtstrafen- Lauf 2
2	35	Hermann Mann	Pro	39,139	4	43,139	2	0	0	0	2	37,509	52	
3	15	Karl Schmidt	Pro	39,148	104	223,148	50	0	0	50	2	34,569	2	
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														
52														
53														
54														
55														
56														
57														
58														
59														
60														
61														
62														
63														
64														
65														
66														
67														
68														
69														
70														
71														
72														
73														
74														
75														
76														
77														
78														
79														
80														
81														
82														
83														
84														
85														
86														
87														
88														
89														
90														
91														
92														
93														
94														
95														
96														
97														
98														
99														
100														

Abbildung 2.6: Ausschnitt aus einer Auswertungsdatei mit Beispieldaten, dargestellt von LibreOffice

2.4 Die Android-App

- Android-App, entwickelt mit Android Studio, der offiziellen IDE von Google
- verwendete Programmiersprache: Java, Interpretation auf Smartphone durch Dalvik VM

- lauffähig ab Android 2.3.3 auf allen Smartphones und Tablets mit WLAN
- läuft im Wettbewerb auf mehreren Geräten parallel

2.4.1 Funktionen

- Tabelle 2.2 zeigt alle Features der App zusammengefasst; bei Interesse an Umsetzung steht Quellcode zur Verfügung, entsprechende Klassen und Methoden sind zugeordnet

Feature	Realisierung mittels
Verbindungsaufbau zur Datenbank über den PHP-Server	Activity ConnectionActivity, Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Abfrage_Stationen.php
Anzeige des aktuellen Laufs	Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Abfrage_Lauf.php
Handstarten eines ausgewählten Starters	Activity StartActivity, Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Abfrage_Startnummern.php, PHP-Skript Startzeit_eintragen.php
Handstoppen eines ausgewählten Starters	Activity StoppActivity, Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Abfrage_Startnummern.php, PHP-Skript Zielzeit_eintragen.php
Anzeigen der vergangenen Zeit seit Wettbewerbsstart	Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Abfrage_Startzeit.php, Klasse Thread, Instanz „uhr“
Auswahl der aktuellen Startnummer und ihren Strafen an den lokalen Toren	Activity MainActivity, Klasse android.widget.Spinner, Klasse android.widget.TableLayout
Eintragen der gewählten Strafzeiten an den gewählten Toren für die gewählte Startnummer	Methode MainActivity.onClick, Klasse HTTP_Connection, Interface AsyncResponse, PHP-Skript Zeit_eintragen.php

Tabelle 2.2: Features der App „Android Connector App HTTP“

2.4.2 Umsetzung des Verbindungsaufbaus

- Android: keine Möglichkeit, direkt auf MySQL-DB auf fremdem Gerät zuzugreifen
- aber: Fähigkeit, GET-Anfragen zu senden und zu empfangen
- Problem dabei: Anfragen dürfen ab Android 3.0 nicht (mit einfachem Code) auf Main- oder UI-Thread ausgeführt werden, siehe hierzu [6]
- Hintergrund: längere Operationen wie Netzwerkzugriffe (mit Verzögerungen der Antwort) würden Thread blockieren und UI einfrieren
- —> Alternative: Klasse AsyncTask beerben (getan in Klasse HTTPConnection) —> Anlegen eines zweiten Threads, in dem die Abfrage durchgeführt werden kann; Thread läuft parallel zum Mainthread

- AsyncTask stellt Methode `doInBackground` bereit, welche auf diesem zweiten Thread läuft
- dort: Abfrage möglich, Klasse `URLConnection` unter Verwendung von `URL.openConnection()` kann nun die übergebene HTTP-Abfrage senden; diese: Anfrage an PHP-Skript auf dem Hauptrechner, zusammengesetzt `http://IP-AdressedesRechners/AndroidConnectorAppHTTPScripts/Skriptname.php?Parameter="Wert"`; Apache-Server nimmt entgegen, führt aus, gibt ein HTML-Document zurück, das ggf. die Ausgaben des Scriptes enthält
- Document kann als `InputStream` von Instanz von `URLConnection` eingelesen werden, wird als String zeilenweise gelesen
- danach: Aufruf der Methode `onPostExecute` im AsyncTask, die im Mainthread läuft; Problem: Antwort muss nun zurück in Activity, das Abfrage ausgeführt hat
- dafür: Interface `AsyncResponse` → enthält Methode `processFinish` → `HTTP_Connection` enthält Instanz des Interfaces, ruft Methode auf
- Interface von Activity implementiert, Überschreibung der leeren Standardmethode → Antwort kann nun in Methode weiter verarbeitet und in Oberflächenelemente geschrieben werden

2.5 Der PHP-Server

- PHP-Server läuft im Apache Webserver, der Teil von XAMPP ist, auf dem zentralen Laptop
- kann durch Anfragen (von den App-Instanzen) nur im lokalen Netzwerk angesprochen werden
- „htdocs“-Ordner enthält erstellte Scripte, welche Anfragen der App verarbeiten, sowie DB-Konfiguration; wird alles von Hauptprogramm vor Start kopiert

2.5.1 Aufgabe

- Entgegennehmen von Anfragen der App, Eintragungen in DB
- Anfragen dabei in Form von GET
- Ausführung von SQL-Befehlen in Datenbank, die entweder einen Schreibzugriff oder eine Leseanfrage durchführen
- bei Leseanfrage: Antwort an App in Form einer HTML-Seite mit den angeforderten Informationen

2.5.2 Umsetzung des Zugriffes auf die Datenbank

- Code-Verweis
- Einlesen der Datei `Konfiguration.php`, welche die Konfiguration in Bezug auf Host, User, Passwort und Datenbankname enthält; wird vom Programm automatisch mit den aktuellen Werten überschrieben

- Aufbau eines Zugriffs
- String zusammensetzen, der den SQL-Befehl enthält mittels Zusammenfüger „.“; übergebene Parameter in der Anfrage können per `$_GET['Parametername']` eingebaut werden (z.B. hier die Nummer der Station)
- Ausführung der Abfrage über `mysqli_query`
- Rückgabe aus der DB „zeilenweise“ durchgehen, Werte aus den nötigen Spalten auslesen und per echo auf die Seite drucken

2.6 Technische Umsetzung der Synchronisation

- Hauptprogramm mit JavaFX geschrieben, kann unter Verwendung des Datenbanktreibers jdbc (externe JAR-Library) und dem `java.sql`-Package direkt auf Datenbank zugreifen
- → leert anfangs die DB, legt Tabellen an, die benötigt werden, trägt Namen und Startnummern in Tabelle „namen“ ein sowie nötige Attribute wie die Zahl der Messstationen sowie die Zurodnung der Messtore zu den Stationen in die Tabelle „allgemein“ → nötig für App, um Startnummernauswahl für Start/Stop oder Anmeldung an Messstation zu realisieren
- Android-Clients mit App in ihren verschiedenen Activitys zum Starten, Stoppen oder auch Strafzeiten eintragen; Android erlaubt keinen direkten MySQL-Zugriff
- → aber: Verbindung zum Server über GET-Anfragen per WLAN über IP und HTTPS möglich (App braucht dafür IP des Servers)
- PHP-Server (hier Apache Webserver, Teil von XAMPP) kann Anfragen entgegennehmen und Abfragen in die DB ausführen
- → App sendet anfangs Anfrage an Server, z.B., um die Zahl der Messstationen zu erfragen
- Server nimmt Anfrage entgegen, führt hier Script „Abfrage_Stationen“ aus
- dieses sendet Anfrage an DB, die als Resultat die Zahl der Messtationen liefert
- PHP-Script gibt Zahl per Befehl „echo“ auf die vom Server zurückgegebene HTML-Seite aus
- App liest Seite per InputStream ein, trennt sie (bei mehreren, vom Server per „|“ getrennten) Anfragen per `String.split` auf und verarbeitet sie, z.B., indem die vorhandenen Messstationen zur Auswahl in einen Spinner gepackt werden
- bei bestimmten Aktionen wie dem Start entfällt die Verarbeitung der Rückgabe (sinnlos), PHP-Server übernimmt Zeiterhebungen bei Start und Stopp (kompensiert Abweichung der Uhren der Smartphones)
- Hauptprogramm → Zusammenführung, Anzeige, Auswertung der Daten der DB und der eigenen, lokalen Daten (z.B. der anderen Starter)

- Programm liest anfangs Starter, Startnummern, Kategorien ein; bereitet DB für die einzelnen Kategorien vor
- Programm startet im vorgegebenen Intervall, standardmäßig jede Zehntelsekunde, die Zeitberechnungsprozedur
- diese ordnet ggf. Startzeiten von der App den Startern zu, markiert sie als gestartet, rechnet die Zeiten entsprechend um (PHP hat eigenes Timestampformat)
- dann: Auslesen der Stoppzeiten aus der DB, die genau wie Startzeiten behandelt werden
- Auslesen, Aufsummieren, Ausgaben der Strafen aus der DB zu Startern
- Programm erkennt automatisch Laufende, sichert ggf. Daten, bietet Möglichkeit an, nächsten Lauf/Kategorie zu starten
- bei Ende letzter Kategorie: Programm erzeugt Excel-Datei mit Auswertung

2.7 Beschreibung des Ablaufs einer Verwendung des Softwaresystems

- zuerst: Öffnen des Hauptprogramms
- erst Start des ConfigWindows, das Starter und weitere Konfigurationsdaten einliest, kann Starter aus Excel-Tabelle auslesen oder hier eingeben
- Klick auf Weiter oder Schließen des Fensters: Frage der Kategorie, die laufen soll; Öffnen des zweiten Fensters, das die Läufe verwaltet
- Start/Stopp von Startern hier möglich, aber unpräzise; Feature in der App enthalten
- Fenster zeigt jederzeit an, wer auf der Strecke ist, dessen aktuelle Zeit mit und ohne Strafen (über Db mit App gesynct) und bei Startern, die im Ziel sind, die Gesamtzeit
- nun: Start der App möglich
- zuerst: Eingabe der IP-Adresse des Hauptrechners im lokalen Netzwerk (auf GUI des Hauptprogramms angezeigt)
- Klick auf „VERBINDEN“
- zum Starten: Klick auf „STARTEN“
- Auswahl der Startnummer im Spinner
- Tipp auf „STARTEN!“
- zum Stoppen: Klick auf „STOPPEN“, selbes Verfahren
- zum Strafzeiteneintragen: Tipp auf die Zahl (Spinner), Wahl der gewünschten Messstation

- Klick auf „WÄHLEN“
- Auswahl der Startnummer, die eine Strafzeit bekommt, per Tipp auf Spinner
- Wahl der Strafzeiten aus Spinner für angegebenes Tor
- Klick auf „Zeit nehmen“
- Zurücksetzen der Anzeige bei anderer Startnummer
- wenn der letzte Starter das Ziel erreicht, werden die Zeiten im Hauptprogramm gespeichert; Möglichkeit, nächsten Lauf zu starten; Start- und Zielzeiten sowie Strafen bis zum Start des nächsten Laufs veränderbar
- ist der nächste Lauf der Wiederholungslauf, wird gefragt, wer antritt
- sonst: Start des regulären zweiten Laufs
- tritt keiner an oder ist der Wiederholungslauf beendet, wird das ConfigWindow wieder aufgerufen, zeigt Zustand vor Kategoriestart
- Klick auf weiter: Auswahl der nächsten Kategorie aus den übrigen Kategorien
- Verwendung der App wieder möglich, kein Neustart erforderlich, nur aktuelle Anzeige durch Kippen des Bildschirms aktualisieren
- alle Kategorien gefahren: Weiterleitung zur Auswertung → Programm speichert Excel-Datei mit Daten an gewähltem Ort

2.8 Aufgetretene Probleme und ihre Lösungen

- gelegentliches Auftreten nicht näher bestimmter Netzwerkfehler, die HTTP-Verbindung zwischen Hauptrechner und App unterbrechen
- → zufällige Fehler, die in Rechnernetzwerken unvermeidbar sind und kaum verhindert werden können
- Besserung durch Wiederholung der Anfrage bis zum Erreichen einer Bestätigung möglich, dabei aber besonders bei Start und Stopp Verzögerung bis zur Wiederherstellung der Verbindung, ggf. mehrere Sekunden → Fehler bei Zeitmessung
- auch: potentielles Problem der WLAN-Reichweite (Streckenlänge in Jena keine 200 m, auf offiziellen Strecken bis 400 m) bei WLAN-n-Spez. von nur 150 m Reichweite
- Verlängerung der Reichweite mit Repeatern möglich, leider keine tragbaren akkubetriebenen Geräte verfügbar, aber gerootete Android-Smartphones können das
- gelegentliche Performanceprobleme bei schwachen Rechnern mit rechenintensiven Nebenthreads

- Ursache: mehrere DB-Zugriffe und komplizierte Zeitenberechnung jede Zehntelsekunde
- Lösung: manuell Synchronisierungstaktrate herunterstellen
- Hauptprogramm kann ohne Unterordner lib mit zusätzlicher Library nicht laufen
- Ursache: DB-Treiber benötigt, Hinzufügen in eigene JAR unter verwendeter IDE NetBeans nicht möglich; Treiber muss immer da sein
- Start von XAMPP per Konsole nur unter Windows möglich, ungelöst
- aufgetretenes Android-internes Problem: nach Neukompilierung der App ohne jegliche Änderung Darstellung der Spinner mit weißer Schrift auf weißem Hintergrund
- gelöst durch Überschreiben eines (meiner Meinung nach) unsinnigen Standard-XML-Designs durch eigenes
- ebenfalls nach Neukompilierung: Absturz der App beim Versuch, Strafzeiten einzutragen
- Ursache: Zugriff auf Spinner und TextViews mit Tornummern in der Tabelle aus unklarem Grund nicht mehr möglich
- —> Speicherung der Adressen der Spinner in Array, Wiederherstellung des Zugriffs, Speicherung der zugehörigen Tore in 2. Array, damit Strafzeiten auch für das richtige Tor eingetragen werden (anfangs ein Problem)
- bei Änderung der Konfigurationswerte im Programm kein Zugriff per App mehr möglich
- Lösung: Überschreibung der PHP-Konfigurationsdatei
- Problem: fehlende PHP-Skripte (vergessen zu kopieren)
- Lösung: Skripte manuell zu Programm kopieren, Programm selbst kopiert diese in XAMPP-Unterverzeichnis
- gelegentliches Problem: beim Konsolenstart von XAMPP bleibt Apache auf manchen PC's hängen
- ungelöst, aber Empfehlung an User, bei Verbindungsproblemen einfach Apache neuzustarten
- Probleme bei Umrechnung von PHP- in Javatimestamps gelöst durch eigene Umrechnungsmethode
- Probleme mit Zeitumrechnung und Differenzbildung gelöst durch Verwendung von java.util.Calendar
- allgemein: verwendete Sprachen Java in seinen Spezifikationen JavaFX und Dalvik unter Android, PHP, MySQL sehr stabil; werfen bei Programmfehlern Exceptions bzw. geben Fehler aus —> Fehler nicht fatal, echte Abstürze sind selten
- letzte Programmversion mit Patches lief stabil

3 Zusammenfassung und Schluss

3.1 Umsetzung der Konzeption

- Softwaresystem kann zur Erfassung von Lauf- und Strafzeiten beim Kanusport verwendet werden, Auswertung ist noch optimierbar, aber definitiv leichter und besser mit der erzeugten Excel-Datei als mit Papierlisten
- Aufbau des Systems ist wie gefordert kostenlos und einfach möglich
- Hardwareanforderungen:
 - 1 Laptop als Hauptserver
 - ggf. Repeater oder als solcher konfigurierter Router (WLAN-n-Spezifikation: 150 m Reichweite, bei abzudeckendem Radius von ca. 100 m reicht 1 Gerät vermutlich aus, um eine ausreichende Abdeckung zu erzielen)
 - für jeden Helfer ein Android-Smartphone mit mind. Android 2.3.3 (Gingerbread)
- Softwareanforderungen
 - Java JRE (auf meisten Geräten sowieso installiert)
 - XAMPP (Installation in wenigen Minuten)
 - Anlegen einer Datenbank (wenige Klicks)
- Einlesen der Starterkonfiguration aus Excel-Tabelle möglich, Starterkategorien frei benennbar (regionale Unterschiede), Programm ordnet Starter automatisch den Läufen zu
- Starten und Stoppen sowie Eintragen von Strafzeiten per App
- automatische Erkennung des Laufendes, Erzeugung von Protokolldateien (ggf.), Datenspeicherung für Auswertung in einer übersichtlichen Excel-Datei
- erzeugte Excel-Datei kann nun in verschiedenen (auch kostenlosen) Programmen wie LibreOffice geöffnet und leicht ausgewertet werden
- korrekte Anzeige der Werte des aktuellen Laufs im Hauptprogramm, kann durch Anschluss an einen Beamer ggf. sogar Zuschauern live gezeigt werden
- System erfordert keinen Internetzugriff -> schwer manipulierbar
- -> alles in allem: Konzeption voll umgesetzt, Problem erfüllt, Programm kann ggf. durch andere Entwickler leicht um weitere Funktionen erweitert werden

3.2 Ausblick

- mögliche Verbesserung: bessere, detailliertere Auswertung; Übergabe an Fremdprogramm (eigens dafür programmiert) in Arbeit
- Livestream der Laufwerte ins Internet durch Auslesen der DB ohne Weiteres möglich, kostengünstig einrichtbar (nur 1 Webserver nötig, der Daten weitergibt; XAMPP für diese Aufgabe nicht zu empfehlen)
- höhere Datensicherheit durch Authkeys für die Skripte und Passwörter für die Stationen zum Schutz vor Manipulation gut möglich, Verschlüsselung der Parameter durch md5
- Verbesserung: Anzeige der aktuell eingetragenen Strafzeiten in den Spinnern der Handy-App; könnte aber Entscheidung des Helfers beeinflussen
- vollständige Integration der benötigten Ressourcen in die JAR (von Ant und NetBeans nicht zugelassen, aber z.B. in Android möglich)
- Erhöhung der Stabilität, Kompensation von Netzwerkfehlern durch Wiederholung der Anfragen bis zur Bestätigung
- Überarbeitung der Fehlerdiagnose und Fehlermeldungen für noch leichtere Identifikation von Fehlern im Ablauf
- Verbesserung des Wiederherstellungsmodus, sodass nicht eine ganze Kategorie wiederholt werden muss
- Abfangen möglicher Eingabefehler bei den Starterdaten, z.B. unzulässige oder doppelte Startnummern
- Auslösen des Start-/Stopp-Signals durch eine Lichtschranke (hochpräzise) -> nur Implementation eines Listeners nötig, der „Start“-Button auf der GUI des Hauptprogramms startet
- Arduino mit WLAN-Modul kann HTTP-Requests senden -> ähnliche Implementierung wie Start durch App möglich, aber startende/stoppende Startnummer müsste durch Helfer eingestellt werden
- über Lichtschranke, Digitalkamera und Arduino auch einfaches Fotofinish möglich, aber hier eher nicht sinnvoll

Glossar

Activity	Entsprechung des Programmfensters unter Android. Eine App kann aus einer oder mehreren Activities bestehen, die sich gegenseitig starten.
Apache POI	„POI ist Open-Source-Software, die Java-APIs zum Lesen und Schreiben von Dateien im Dateiformat von Microsoft Office wie z. B. Word und Excel bereitstellt. “[12]
GET-Anfragen	Anfragen an Scripte eines PHP-Servers. Parameter werden dabei in der Webadresse übergeben. Nach dem Namen des aufzurufenden Scriptes, etwa <code>http://192.168.1.234/Zeit_eintragen.php</code> , stehen die Parameter in der Form <code>?station=0&starter=1&strafe=50</code> .
htdocs-Ordner	„HyperText Documents“, Ordner in XAMPP. Dort liegen alle Skripte und HTML-Seiten, die über den Webserver aufgerufen werden können.
Hypertext Transfer Protocol	Netzwerkprotokoll, das für die Übertragung von Anfragen an Server und Antworten an Clients in Form von Hypertext (HTML) entwickelt wurde.
IP	Protokoll, das Anfragen und Antworten im World Wide Web zu bestimmten Servern oder Clients lenkt, die über eine IP-Adresse eindeutig identifiziert sind.
MariaDB	„freies, relationales Open-Source-Datenbankverwaltungssystem, das durch eine Abspaltung (Fork) aus MySQL entstanden ist“[10]
MySQL	Sprache, die zur Erzeugung und Anlegung von Datenbanken genutzt wird.
PHP	Abkürzung von Hypertext PreProcessor. Scriptsprache, die hauptsächlich auf Servern ausgeführt wird. Nimmt eine POST- oder GET-Request entgegen, verarbeitet sie z.B. mit einem Zugriff auf eine MySQL-Datenbank und gibt eine HTML-Seite an den anfragenden Client zurück.
UI-Thread/Mainthread	Thread, in dem die eigentlichen Activities der App ausgeführt werden. Jede App erhält bei Start einen solchen Thread zugewiesen. Seine Hauptaufgabe ist die Verknüpfung von Oberflächenelementen mit dem Programmcode, der ihren verschiedenen Events zugewiesen ist. Außerdem ist er für die Modifikation der Oberflächenelemente wie neue Texte in einem TextView verantwortlich.
XAMPP	„XAMPP ermöglicht das einfache Installieren und Konfigurieren des Web-servers Apache mit der Datenbank MariaDB bzw. SQLite und den Skript-sprachen Perl und PHP (mit PEAR). “[11]

Literatur

- [1] CANOPUS49: *Wi-Fi icon in pale blue*. https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/WIFI_icon.svg/2000px-WIFI_icon.svg.png. Version: 26.12.2016 17:53
- [2] GOOGLE INC.: *Android Robot*. https://upload.wikimedia.org/wikipedia/commons/thumb/d/db/Angry_robot.svg/2000px-Angry_robot.svg.png. Version: 26.12.2016 18:23
- [3] IMAS: *Basissystem*. <http://www.imas-sport.com/index.php/de/zeitmessung/basissystem>. Version: 26.12.2016 15:53
- [4] IMAS: *Komplettsystem*. <http://www.imas-sport.com/index.php/de/zeitmessung/komplettsystem>. Version: 26.12.2016 16:00
- [5] IMAS: *Versionen im Vergleich*. <http://www.imas-sport.com/index.php/de/zeitmessung>. Version: 26.12.2016 17:54
- [6] LOCKWOOD, Alex: *Why Ice Cream Sandwich Crashes your App*. <http://www.androiddesignpatterns.com/2012/06/app-force-close-honeycomb-ics.html>. Version: 28.12.2016 15:56
- [7] PIXABAY: *Android Smartphone Clipart*. https://cdn.pixabay.com/photo/2016/03/31/19/26/android-1295022_960_720.png. Version: 26.12.2016 17:48
- [8] PROTIME SPORTSERVICE GMBH: *Preisliste*. <http://www.zeitmessung.de/preisliste.html>. Version: 26.12.2016 16:53
- [9] WALTER, Stéphanie: *Mobile and desktop device template*. https://upload.wikimedia.org/wikipedia/commons/d/d4/Devicetemplates_laptop-01.png. Version: 26.12.2016 17:44
- [10] WIKIPEDIA: *MariaDB*. <https://de.wikipedia.org/wiki/MariaDB>. Version: 26.12.2016 18:33
- [11] WIKIPEDIA: *XAMPP*. <https://de.wikipedia.org/wiki/XAMPP>. Version: 26.12.2016 18:37
- [12] WIKIPEDIA: *Apache POI*. https://de.wikipedia.org/wiki/Apache_POI. Version: 31.12.2016 17:41

Alle Quellen, die für die Softwareentwicklung relevant waren, sind im **@author**-Tag der Javadocs oder an verwendeter Stelle in Programmkomentaren gekennzeichnet.

4 Anhang

Bilder, Struktogramme, evtl. Code

4.1 Funktionsweise der Hauptmethode, welche alle Daten zusammenführt

- Struktogramm oder Pseudocode mit kurzer Erläuterung
- *Anhang*

4.2 Protokollierung und Wiederherstellungsmodus

- kurzes Statement, wie Dateien gespeichert und wieder gelesen werden
- *Anhang*

4.3 Screenshots

4.3.1 Hauptprogramm

Startnummer	Name	Kategorie
1	Max Mustermann	Elite
3	Franz Schuster	Elite
15	Karl Schmidt	Pro
27	Franz Franz	Ober-Elite
35	Herrmann Mann	Pro
10	Hans-Dietrich Genscher	Ober-Elite

aus Datei lesen neue Zeile Zeile entfernen Startnummern automatisch generieren

Bitte geben Sie hier die Zahl der Messstationen ein! Bitte geben Sie hier die Zahl der Messtore ein!

Bitte geben Sie hier die Konfigurationswerte des MySQL-Servers ein, sofern Sie diese verändert haben!

Hostname: Port:

Datenbank: ☒ Datenbank automatisch leeren?

Benutzername: Passwort:

Speicherort von XAMPP:

☒ Netzwerkverbindung vor Programmstart prüfen?

☒ Sicherungsprotokolle aktivieren?

Speicherort:

Abbildung 4.1: Konfigurationsfenster des Hauptprogramms mit Beispieleingaben

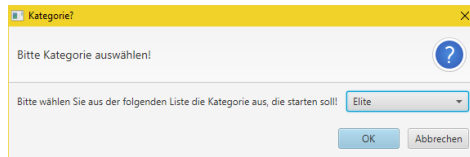


Abbildung 4.2: Wahl der Kategorie, welche starten soll

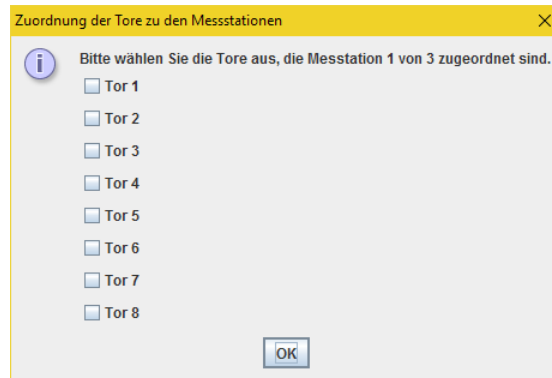


Abbildung 4.3: Zuordnung der Messtore zu Messstation 1

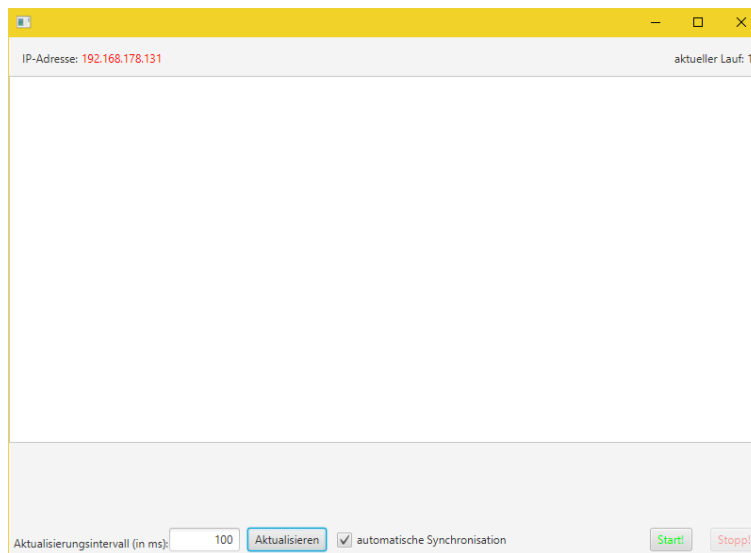


Abbildung 4.4: Hauptfenster vor Start des ersten Starters

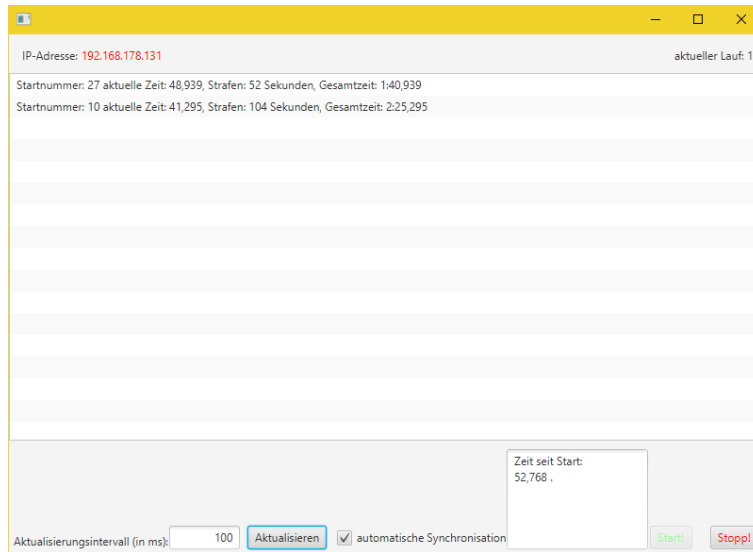


Abbildung 4.5: Hauptfenster während eines Laufs

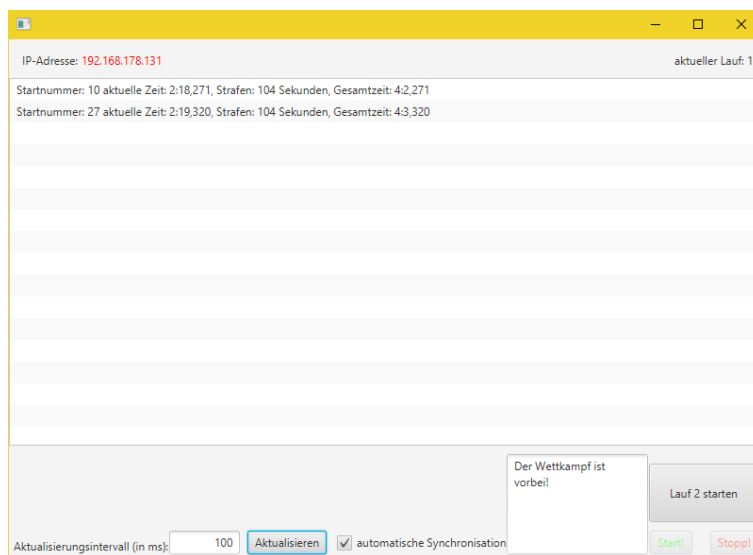


Abbildung 4.6: Hauptfenster bei Laufende

4.3.2 App

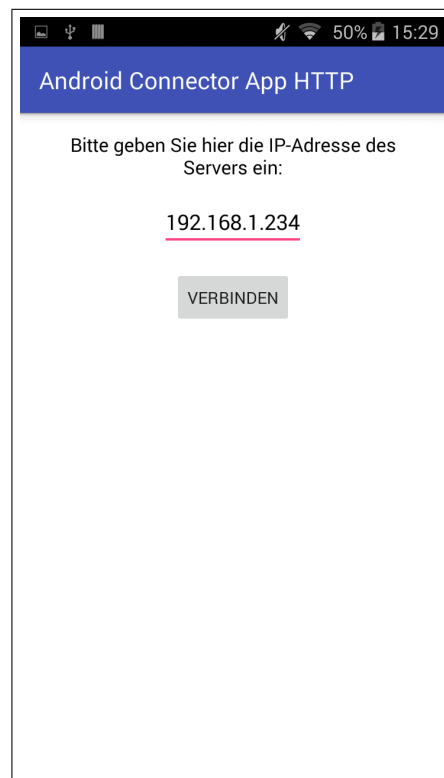


Abbildung 4.7: App vor dem Verbindungsaufbau

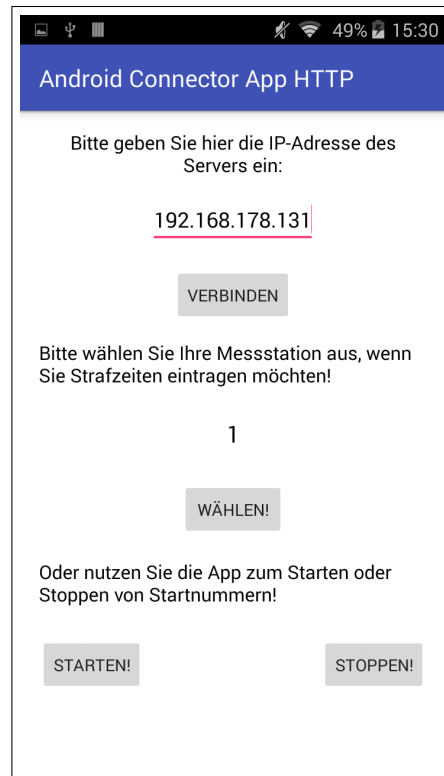


Abbildung 4.8: App bei der Funktionswahl

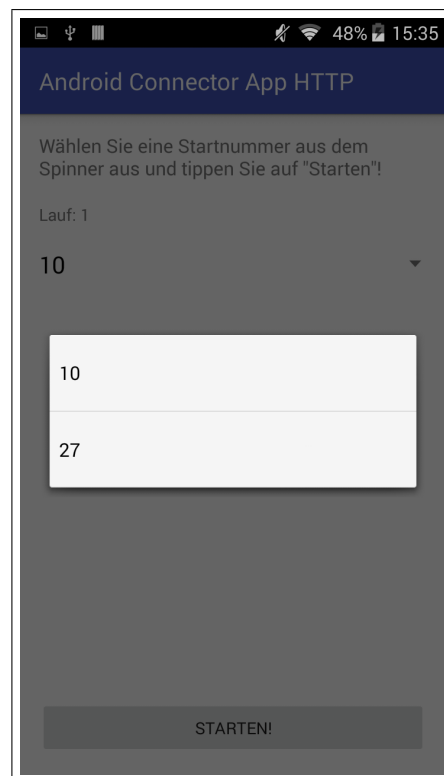


Abbildung 4.9: App im Startmodus bei Auswahl der Startnummer (Stoppmodus ähnlich)

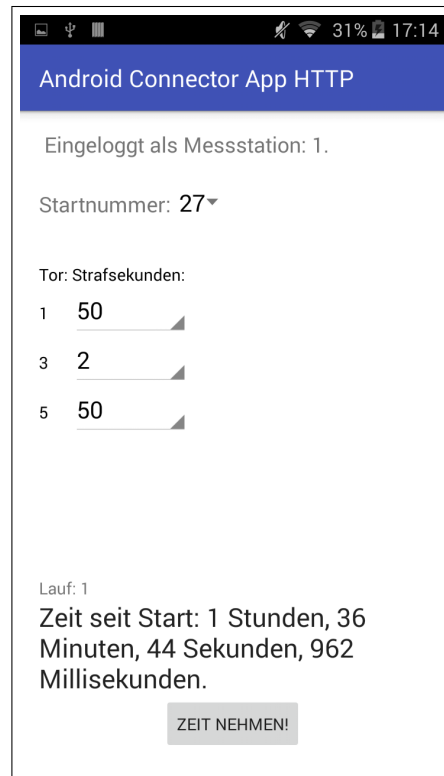


Abbildung 4.10: App beim Eintragen einer Strafzeit

4.3.3 Protokolle

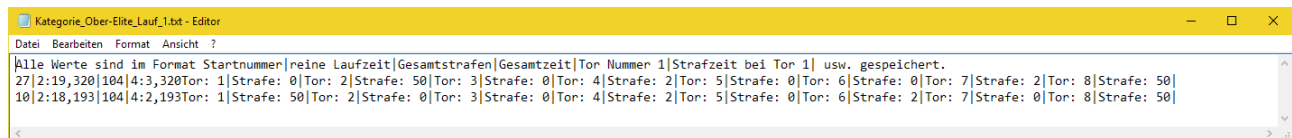


Abbildung 4.11: Protokoll eines Programmlaufs mit Beispielausgaben

4.4 Sourcecode

```

1 /**
2  * Hauptkonstruktor der Klasse
3  *
4  * @param host Datenbankhost (Standard: "localhost")
5  * @param port Datenbankport (Standard: "3306")
6  * @param db Datenbankname (Standard: "android_connect")
7  * @param user Datenbankbenutzername (Standard: "root")
8  * @param password Datenbankpasswort (Standard: leer)
9  */
10 MySqlConnection(String host, String port, String db, String user, String password,
    FXMLDocumentController doc) {
11     try {
12         //Verbindungsdaten speichern
13         this.dbHost = host;
14         this.dbPort = port;
15         this.database = db;

```

```

16         this.dbUser = user;
17         this.dbPassword = password;
18         // Datenbanktreiber für ODBC Schnittstellen aus der Library laden.
19         // Resultat: im java.sql.DriverManager wird (statisch) der geladene Treiber
           als Datenbanktreiber hinterlegt, dafür verantwortlich: Befehl im static-
           Teil der Treiberdefinition, der bei Zugriff auf die Klasse ausgeführt
           wird
20         // Treiber wird nur gefunden, wenn lib-Unterordner da ist!
21         Class.forName("com.mysql.jdbc.Driver");
22         // Verbindung zur ODBC-Datenbank android_connect herstellen und für
           Verwendung speichern.
23         // Es wird die JDBC-ODBC-Brücke verwendet.
24         conn = DriverManager.getConnection("jdbc:mysql://" + dbHost + ":" + dbPort +
           "/" + database + "?" + "user=" + dbUser + "&" + "password=" + dbPassword
           );
25         this.doc = doc;
26     } catch (ClassNotFoundException e) {
27         // aufgerufen, wenn Treiber nicht gefunden wird
28     } catch (SQLException e) {
29         // aufgerufen, wenn keine Verbindung zur DB möglich
30     }
31 }

```

Listing 1: Hauptkonstruktor der Klasse „MySQLConnection“, welche eine Verbindung zur Datenbank herstellt

```

1 // Konfiguration einlesen
2 require_once("Konfiguration.php");
3 // Datenbankverbindung aufbauen
4 $connection = mysqli_connect($host,$user,$password,$datenbank);
5 // Tore abfragen
6 selectTore($connection);
7
8 // liest alle der übergebenen Station zugeordneten Tore aus der Datenbank ein und gibt diese
   aus
9 function selectTore ($connection) {
10     // Abfrage zum Auslesen aller Tore
11     $sqlStmt = "SELECT Wert FROM 'allgemein' WHERE 'Attribut' = 'Messstation_'. $_GET["
           station"]. "_Tore" ";
12     // Abfrage ausführen, Resultat in Variable result schreiben
13     $result = mysqli_query($connection,$sqlStmt);
14     // falls Abfrage erfolgreich...
15     if ($result = $connection->query($sqlStmt)) {
16         // für jede Zeile der Datenbankausgabe (nur 1 Zeile möglich, da Attribut Prim
           ärschlüssel ist)...
17         while ($row = $result->fetch_assoc()) {
18             // die Tore ermitteln...
19             $id = $row["Wert"];
20             // und ausgeben.
21             echo $id;
22         }
23         // Ergebnisse leeren $result->free();
24         // Verbindung schließen
25         closeConnection($connection);
26     }

```

27 }

Listing 2: Methode „selectTore“ des Skriptes „Abfrage_Tore.php“, welche alle Tore ermittelt, die der übergebenen Messstation zugeordnet sind.

Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit mit dem Titel „Wettkampferfassung und -auswertung beim Kanusport mittels eines kostenlos und einfach einrichtbaren Softwaresystems“ selbstständig, ohne unzulässige fremde Hilfe angefertigt und nur unter Verwendung der angegebenen Literatur und Hilfsmittel verfasst habe. Sämtliche Stellen, die wörtlich oder inhaltlich anderen Werken entnommen sind, wurden unter Angabe der Quellen als Entlehnung kenntlich gemacht. Dies trifft besonders auch auf Quellen aus dem Internet zu. Gleichzeitig gebe ich das Einverständnis, meine Arbeit mittels einer Plagiatssoftware überprüfen zu lassen.

Jena, 5.10.2016

Eric Ackermann