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

28. Dezember 2016

betreut durch:
Herrn Bernd Schade

Kurzfassung

Zeitmesssysteme für den Kanusport sind teuer und schwer zu installieren sowie 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 Wettbewerben und Testläufen für diese.
Ziel dieses Projekts ist die Entwicklung eines Softwaresystems, welches dieses Problem vereinfacht. Im Mittelpunkt steht die Entwicklung einer App, welche 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

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung (max. 2 Seiten)

1.1 Stand der Technik und Problembeschreibung ($^3/_4$Seiten)

- Notwendigkeit der Messung und Zusammenführung von Zeiten beim Kanusport (mit Regelwerk unterlegen!)
- Kanuvereine oder Ausrichter von Wettkämpfen müssen bei diesen Laufzeiten und Strafzeiten erheben, zuordnen zu Starter, Kategorie und Lauf, vergleichen...
- Existenz von Systemen zur automatischen Zeiterfassung
- Hersteller etwa IMAS oder Lynx
- vollwertige Anlage z.B. mit „Time System Professional“[4, 5] $\longrightarrow$ 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
- Preis auf Anfrage, da optionale kostenpflichtige Zusatzgeräte wie eine Liveanzeige am Wettbewerbsort
- 6 Laptops benötigt mit jew. 1 Bediener
- Vergleichbares System von Lynx kostet aber bis zu 41.595,- € zzgl. MwSt, in der günstigsten Variante noch 11.995,- € zzgl. MwSt[8]; ebenfalls keine Zusynchronisation von Strafzeiten
- $\longrightarrow$Zeitmesssysteme für Kanusport sind einige vorhanden und bieten viele Funktionen, etwa Start durch Magnetschleifen, Fotoauswertung und Livelisten im Internet
- aber: teuer, einfachste Systeme kosten mehrere 1000 €, Fehlen der Eintragung von Strafzeiten in fast allen Systemen
- $\longrightarrow$zu teuer für kleinen Verein, trotzdem Auswertung mit ggf. 100 Startern nötig
- bisher: Aufschreiben von Strafen per Hand, Auswertung über Excel-Makros
- $\longrightarrow$Zeitaufwand, Organisationsaufwand, Fehleranfälligkeit
- im Rahmen der Arbeit: Versuch, das Problem mit kostenloser, selbst entwickelter Software und günstiger Hardware zu lösen

1.2 Problembeschreibung und Konzeption (1 Seite)

- Grundidee: Softwaresystem, das Aufgabe der Erfassung und Auswertung von Lauf- und Strafzeiten automatisch übernimmt
- Wettkampfstrecke im Verein in Jena ca. 200 m zwischen Start und Ziel
- auf Strecke ca. 18-25 Tore, die durchfahren werden müssen
- möglich dabei: fehlerfreie Durchfahrt (0 Strafsekunden), Berührung des Torpfostens (2 Strafsekunden), Auslassen des Tores oder falsche Richtung (50 Strafsekunden)
- nicht immer alle Tore von einem Punkt aus einsehbar $\longrightarrow$ Unterteilung der Strecke in 4-8 Messstationen; in jeder sitzt Helfer, der für zugewiesene Tore die Strafzeiten der Starter aufschreibt
- Anforderung an System: Handy-App, die jeder Helfer installiert hat
- Helfer wählt Startnummer in App aus und trägt Strafzeiten bei seinen Toren ein, Daten werden automatisch mit Hauptprogramm synchronisiert
- zudem: Zeit der Starter von Start zu Stopp muss gemessen werden
- Helfer an Start- und Ziellinie hat je Handy mit der App
- App ermöglicht Starten oder Stoppen des gewählten Starters per Tipp, Zeitpunkte werden mit Hauptprogramm synchronisiert
- Hauptprogramm legt zunächst möglichst automatisch benötigte Infrastruktur (z.B. Datenbanken) an, bereitet Verbindung vor
- Starter sind in Kategorien (ca. 27) einsortiert, die jeweils in 1 Rennen gegeneinander antreten
- Software liest zunächst Excel-Datei mit Startern und Zuordnung ein, bereitet Erfassung vor
- Bediener wählt Kategorie aus, die startet $\longrightarrow$ Wechsel zum entsprechenden Fenster
- 2 Läufe für jeden Starter, besserer zählt, ggf. Wiederholungslauf für einzelne
- $\longrightarrow$ aus per App übergebenen Start- und Stoppzeiten sowie Strafen wird jederzeit die aktuelle Zeit für jeden Starter berechnet und zur Information angezeigt, bei laufende: Speicherung der Daten für Auswertung
- 2. Lauf hier wie 1. Lauf
- für Wiederholungslauf wird nun ausgewählt, wer startet, dann nach selbem Prinzip weiter
- nach Ende des Wiederholungslauf: $2 + 1$ Läufe für nächste Kategorie usw.
- sind alle Läufe aller Kategorien fertig, Auswertung der Zeiten z.B. als Excel-Tabelle
- Sync per WLAN (lok.) $\longrightarrow$ Schutz vor Fremdzugriff über das Internet
- Verwendung ausschließlich von freier Software; WLAN-Netzwerk mit älteren Geräten kostengünstig aufbaubar

2 Vorgehensweise (ca. 10 Seiten)

2.1 Grundprinzip der Vernetzung der verschiedenen Geräte (1-2 Seiten)

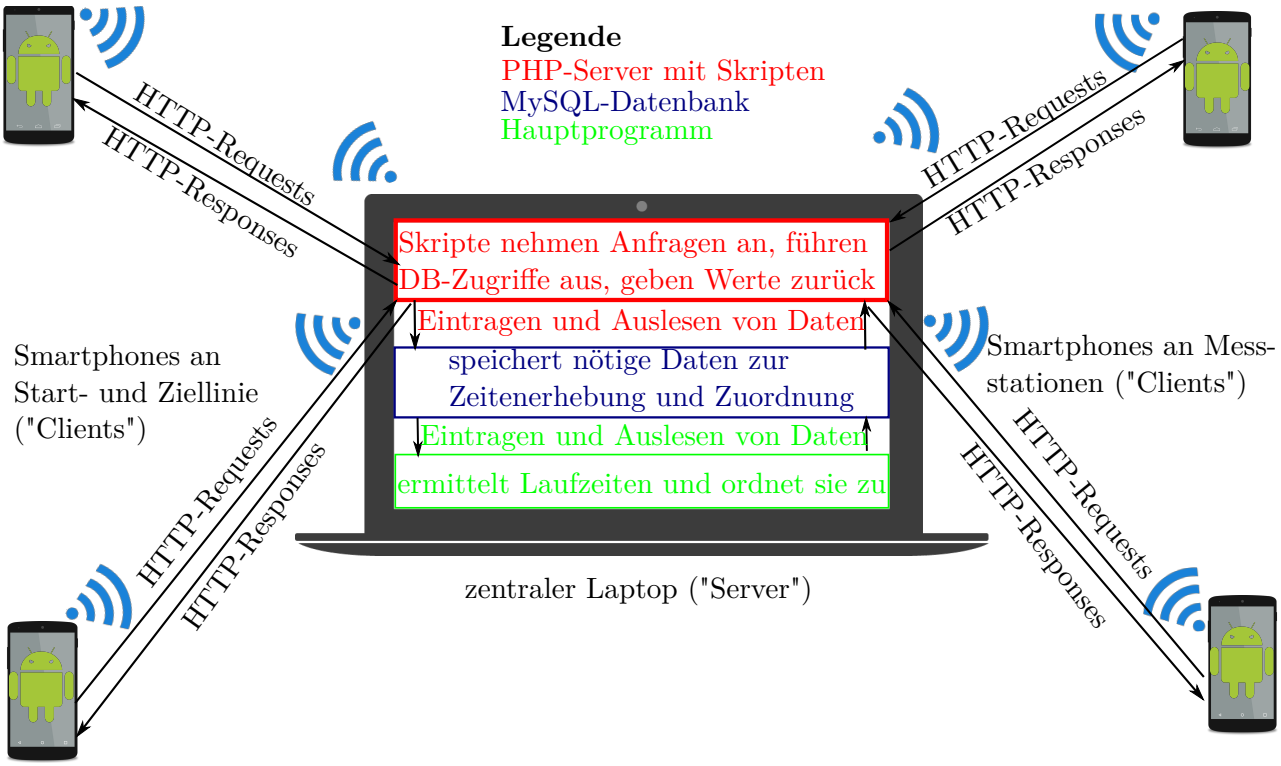


Abbildung 2.1: Schema der Vernetzung der verschiedenen Geräte, vgl. [2, 9, 7, 1, (Illustrationen)]

- in der Mitte (siehe Abbildung 2.1) steht MySQL-DB, die alle Daten speichert
- DB enthält Tabellen, die verschiedene Werte aufnehmen; z.B. Tabelle „Namen“, welche zu jedem Zeitpunkt für die aktuelle Kategorie die Startnummern, zugeordneten Namen und jeweiligen Laufergebnisse speichert
- Datenbank wird durch MariaDB verwaltet (freier Ableger von MySQL mit gleicher technischen Grundlage)
- MariaDB kann nur auf Server ausgeführt werden $\longrightarrow$ zentraler Laptop, der hier auch als Server fungiert, wird zum Anlegen eines Webservers genutzt
- dafür: Installation der Software XAMPP $\longrightarrow$ legt Webserver an, der 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 $\longrightarrow$ evtl. Sicherheitslücken
- Software muss vom Benutzer anfangs installiert und eingerichtet werden (Aufwand ca. 5 Minuten, XAMPP ist für schnelle und einfache Installation gedacht, Installation und Einrichtung der Einzelkomponenten deutlich zeitaufwendiger), wird danach per Systemkommandozeile vom Hauptprogramm gestartet; Programm übernimmt nun automatisch das Kopieren der nötigen Skripte, DB muss einmalig von User angelegt werden

- Hauptprogramm mit JavaFX geschrieben, kann unter Verwendung des Datenbanktreibers jdbc (externe JAR-Library) und dem java.sql-Package direkt auf Datenbank zugreifen
- $\longrightarrow$ 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“ $\longrightarrow$ nötigt für App, um Startnummernauswahl für Start/Stopp 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
- $\longrightarrow$ 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
- $\longrightarrow$ App sendet anfangs Anfrage an Server, z.B., um die Zahl der Messstationen zu erfragen
- Server nimmt Anfrage entgegen, führt 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 $\longrightarrow$ 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

2.2 Aufbau und Funktionsweise der verwendeten MySQL-Datenbank (max. 1 Seite)

- DB (siehe Abbildung 2.2) enthält genannte Tabellen: „namen“, „allgemein“, für jede Messstation 1, sowie evtl. eine für die Auswertung

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 $\longrightarrow$ 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

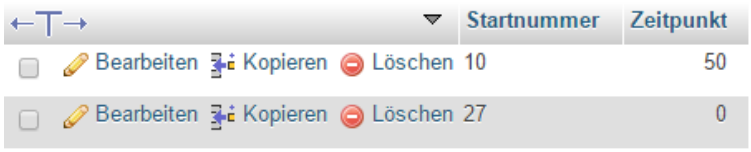
Abbildung 2.3: Beispielbelegung der Datenbanktabelle „allgemein“

- 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
Bearbeiten Kopieren Löschen	Hans-Dietrich Genscher	10	1482847027.7587	1482847166.0294	2:18,193|104|4:2,193Tor: 1|Strafe: 50|Tor: 2|Straf...	NULL	NULL	NULL
Bearbeiten Kopieren Löschen	Franz Franz	27	1482847020.1146	1482847159.4347	2:19,320|104|4:3,320Tor: 1|Strafe: 0|Tor: 2|Strafe...	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
Bearbeiten Kopieren Löschen	10	50
Bearbeiten Kopieren Löschen	27	0

Abbildung 2.5

- Startnummer in allen genannten Tabellen der Primärschlüssel; *mehr zur Technik/DB-Struktur(z.B. Nullwerte, Datentypen?)*

2.3 Das Hauptprogramm (4 Seiten)

2.3.1 Features

- Programmablauf:
  - erst Start des ConfigWindows, das Starter und weitere Konfigurationsdaten einliest
  - 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
  - wenn der letzte Starter das Ziel erreicht, werden die Zeiten 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
  - tritt keiner an oder ist der Wiederholungslauf beendet, wird das ConfigWindow wieder aufgerufen, zeigt Zustand vor Kategoriestart
  - neue Starter können hinzugefügt, noch nicht gelaufene gelöscht werden, neue Kategorien nicht möglich
  - Klick auf weiter: Auswahl der Kategorie aus den übrigen Kategorien
  - alle Kategorien gefahren: Weiterleitung zur Auswertung
- 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 ApacheP 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

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

```
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

- 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 Verwaltung weiterer Daten

- Gesamtheit aller Starterdaten bleibt während des Programmablaufs in javafx.collections.ObservableList namens PersonData gespeichert, die der Tabelle im ConfigWindow zugeordnet ist, Klasse Person repräsentiert dabei die Starter
- Konfigurationsdaten werden in Instanz von ConfigWindowController gespeichert, die beim Wechsel der Anzeige behalten und bei erneutem Aufruf nach Kategorieende wieder gelesen werden, um ConfigWindow wiederherstellen zu können
- zudem: Übergabe der noch nicht gestarteten Kategorien über die Instanz, damit Programm „weiß“, welche Kategorien noch zu starten sind, wenn das ConfigWindow wieserhergestellt wird
- Start- und Stoppzeiten, welche über die Buttons erhoben wurden, sind nur lokal in einem Array von Instanzen von java.util.Calendar gespeichert; ausgelesene Startzeiten aus der DB von der App werden ebenfalls dazu konvertiert –> Programm „weiß“, wer noch nicht gestartet und wer schon fertig ist; schneller, als bei jedem Starter immer die Startzeiten aus der DB auslesen und umrechnen zu müssen
- lokales zweidimensionales int-Array im FXMLDocumentController, das für jedes Tor (1. Index) und jeden Starter (2. Index) die Strafen an den Toren speichert –> Beschleunigung der Auswertung

- Fenster enthalten meist auch Instanzen der anderen Laufklassen sowie des Programmfensters (alle Fenster teilen 1 Stage), um Werte übergeben, ggs. Methoden aufrufen oder Aktionen auslösen zu können

2.3.4 Datenverarbeitung und Auswertung

- Erzeugung einer Zwischendatei, Übergabe an Fremdprogramm
- *noch offen*

2.4 Die Android-App (1 Seite)

2.4.1 Funktionen

- Programmablauf
  - Start der App
  - 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

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
- $\longrightarrow$ Alternative: Klasse AsyncTask beerben (getan in Klasse HTTPConnection) $\longrightarrow$ 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 $\longrightarrow$ enthält Methode processFinish $\longrightarrow$ HTTP_Connection enthält Instanz des Interfaces, ruft Methode auf

- Interface von Activity implementiert, Überschreibung der leeren Standardmethode $\longrightarrow$ Antwort kann nun in Methode weiter verarbeitet und in Oberflächenelemente geschrieben werden

2.5 Der PHP-Server (1 Seite)

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

2.5.2 *Die Scripte und ihre Aufgaben (in Handytabellen bereits erwähnt)*

- *Tabelle über alle Scripte*
- *Erwähnung des Funktionsprinzips an nötiger Stelle, z.B., dass immer die Systemzeit des Hauptprogrammrechners Grundlage für Zeiteintragungen ist (um etwa falsch gehende handyuhren zu kompensieren)*

2.5.3 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

```
1  //Konfiguration einlesen
2  require_once("Konfiguration.php");
3  //Datenbankverbindung aufbauen
4  $connection = mysqli_connect($host,$user,$passwort,$datenbank);
5  //Tore abfragen
6  selectTore($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 Primärschlüssel
                        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.

2.6 Aufgetretene Probleme und ihre Lösungen (1-2 Seiten)

- gelegentliches Auftreten nicht näher bestimmter Netzwerkfehler, die HTTP-Verbindung zwischen Hauptrechner und App unterbrechen
- $\longrightarrow$ 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 $\longrightarrow$ Fehler bei Zeitmessung
- 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
- $\longrightarrow$ Speicherung der Adressen der Spinner in Array, Wiederherstellung des Zugriffs, Speicherung dern 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 $\longrightarrow$ Fehler nicht fatal, echte Abstürze sind selten
- letzte Programmversion mit Patches lief stabil

3 Zusammenfassung und Schluss (max. 2 Seiten)

3.1 Umsetzung der Konzeption (ca. $^3/_2$ Seiten)

- Softwaresystem kann zur Erfassung von Lauf- und Strafzeiten beim Kanusport verwendet werden, Auswertung ist noch in Arbeit, aber voraussichtlich vor Regionalwettbewerb abgeschlossen
- manuelles Verarbeiten der Laufprotokolle stellt bereits einen gangbaren Weg der Auswertung dar, vereinfacht den Aufwand erheblich
- es müssen aus den Protokollen nur noch die gewünschten Informationen kopiert und in eine Excel-Tabelle eingetragen werden; jede erdenkliche weitere Auswertung ebenfalls möglich *Anmerkung: ich erstelle zunächst eine provisorische Excel-Auswertung*
- Aufbau des Systems ist wie gefordert kostenlos und einfach möglich
- Hardwareanforderungen:
  - 1 Laptop als Hauptserver
  - 1 Router, ggf. 1 Repeater (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 variabel, 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
- korrekte Anzeige der Werte des aktuellen Laufs im Hauptprogramm
- System erfordert keinen Internetzugriff –> schwer manipulierbar
- –> alles in allem: Konzeption voll umgesetzt, problem erfüllt

3.2 Ausblick (ca. $^1/_2$ Seite)

- mögliche Verbesserung: bessere, detailliertere Auswertung
- 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
- Erhöhung der Stabilität, Kompensation von Netzwerkfehlern durch Wiederholung der Anfragen bis zur Bestätigung
- Überarbeitung der Fehlerdiagnose 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

Glossar

Activity	Entsprechung des Programmfensters unter Android. Eine App kann aus einer oder mehreren Activities bestehen, die sich gegenseitig starten.
GET-Anfragen	Anfragen an Scripte eines PHP-Servers. Parameter werden dabei in der Webadresse übergeben. Nach dem Namen des aufzurufenden Scriptes, etwa `http://192.168.1.234/Zeit_eintragen.php`, stehen die Parameter in der Form `?station=0&starter=1&strafe=50`.
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 Webservers Apache mit der Datenbank MariaDB bzw. SQLite und den Skriptsprachen 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

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

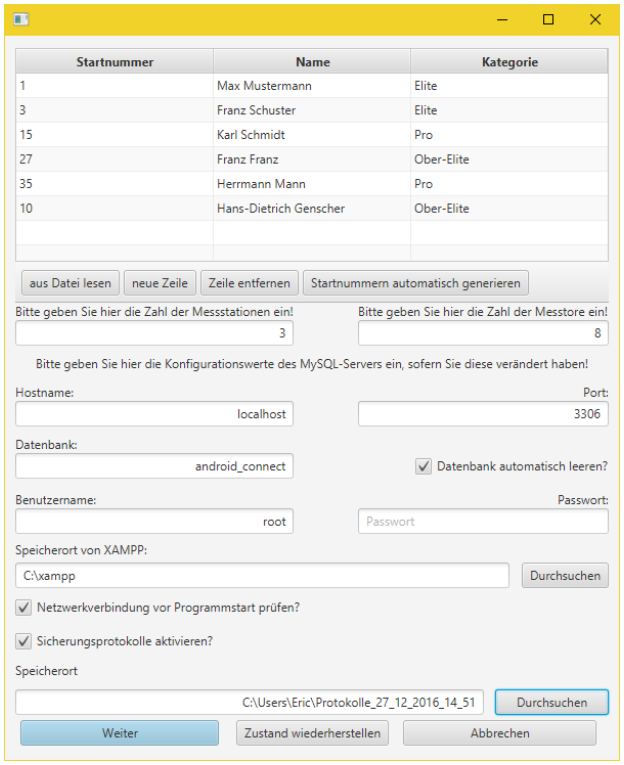


Abbildung 4.1: Konfigurationsfenster des Hauptprogramms mit Beispieleingaben

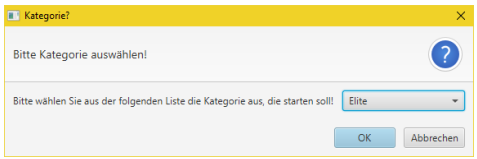


Abbildung 4.2: Wahl der Kategorie, welche starten soll

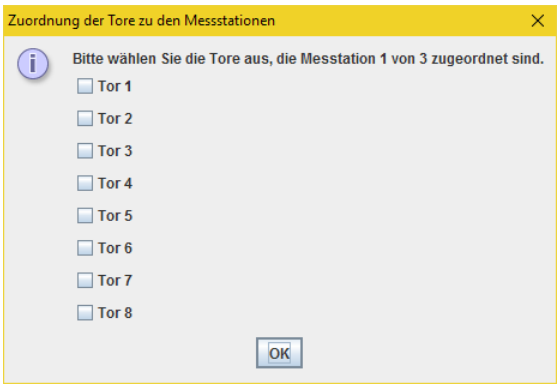


Abbildung 4.3: Zuordnung der Messtore zu Messstation 1

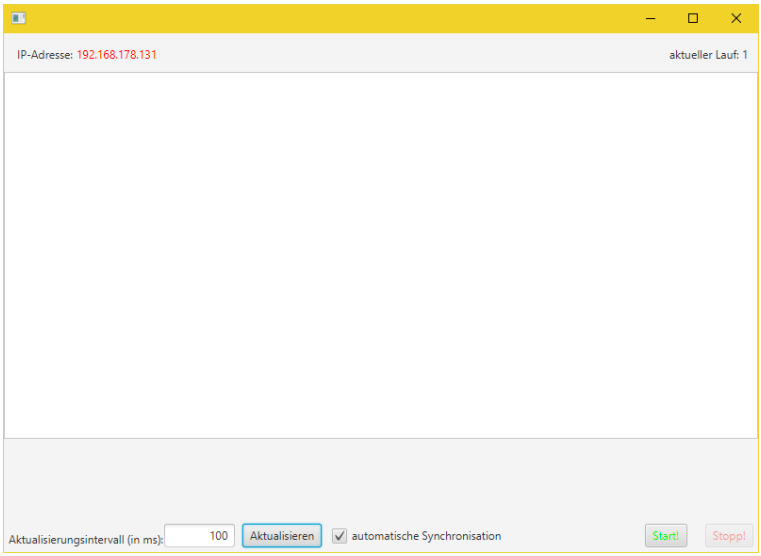


Abbildung 4.4: Hauptfenster vor Start des ersten Starters

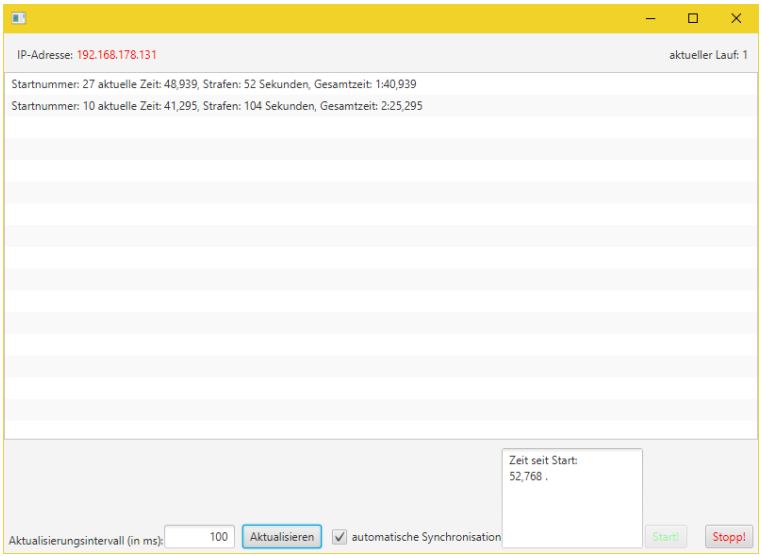


Abbildung 4.5: Hauptfenster während eines Laufs

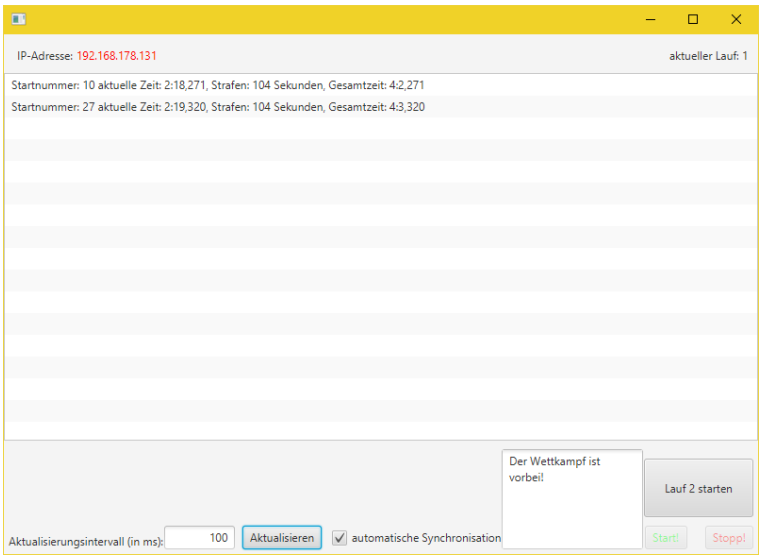


Abbildung 4.6: Hauptfenster bei Laufende

4.3.2 App

Abbildung 4.7: App vor dem Verbindungsaufbau

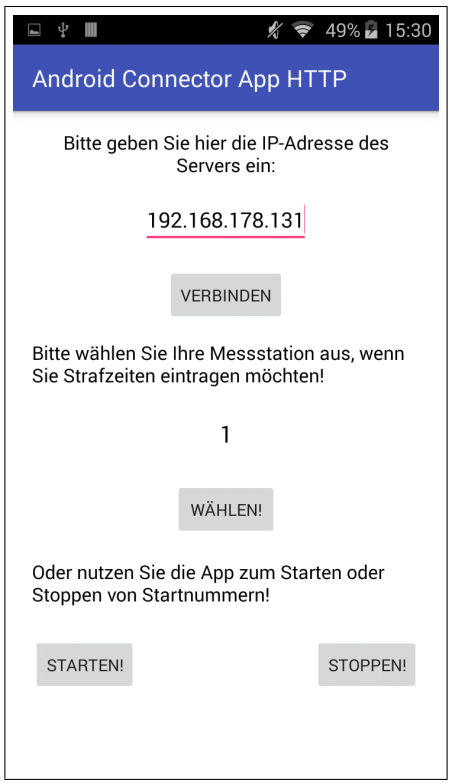


Abbildung 4.8: App bei der Funktionswahl

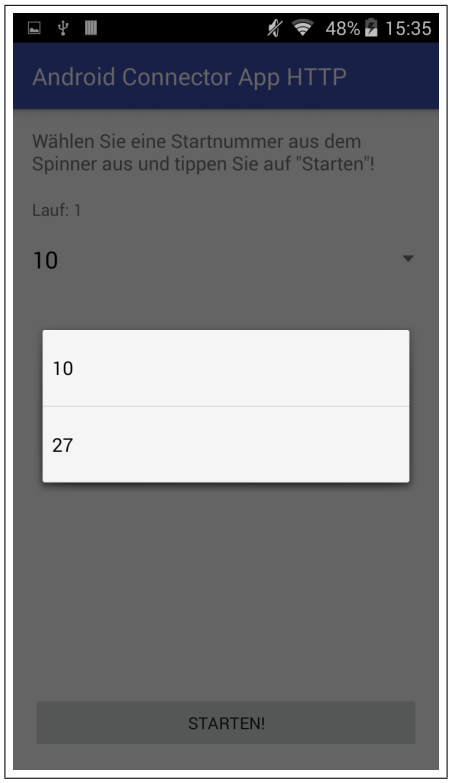


Abbildung 4.9: App im Startmodus bei Auswahl der Startnummer (Stoppmodus ähnlich)

Abbildung 4.10: App beim Eintragen einer Strafzeit

4.3.3 Protokolle

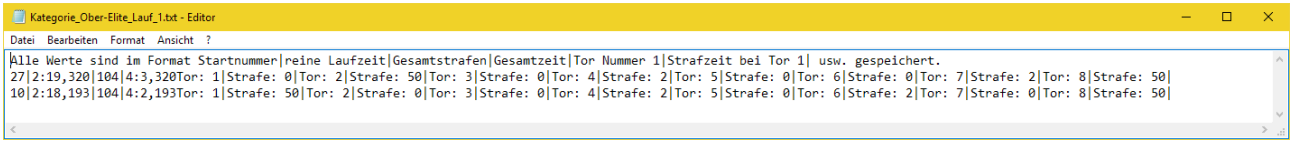


Abbildung 4.11: Protokoll eines Programmlaufs mit Beispielausgaben

Eidesstattliche Erklärung (nicht gefordert)

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