# Konzept - GPS-Tracker

Im Rahmen des Praktikum Software Engineering soll eine Applikation zur Verwaltung von „GPS-Tracks“ erstellt werden. Das Ziel dieser Applikation ist es Nutzern zu ermöglichen ihre aufgezeichneten “GPS-Tracks” zu organisieren, anzusehen und auszuwerten.

Die Applikation wird dabei in drei Sprints umgesetzt und soll spätestens am 19. Juli 2022 vollständig umgesetzt werden. Der Auftraggeber des Projektes ist DI Dr. Johannes Sametinger und das Team zur Umsetzung bilden Ozan Akkoyun, Nuray Seker und Natalie Punz.

Die Applikation ist für sportbegeisterte Personen vorgesehen, welche ihre Aktivitäten elektronisch (ergo in einem der unterstützten Dateiformate) aufzeichnen und in einem Programm verwalten möchten. Die Nutzung der Applikation ist als “Single User”-Applikation bestimmt, weshalb nur ein Nutzer pro Applikationsinstallation gedacht ist. Zur Umsetzung soll in der Programmiersprache Java eine entsprechende Applikation entwickelt, getestet und schließlich veröffentlicht werden.

## Funktionale Anforderungen

| Nr. | Kategorie | Name | Anforderung |
|---|---|---|---|
| 1. | Einlesen | Allgemeines zum Einlesen | Die Applikation ermöglicht das Einlesen von Dateien im Format “.tcx” und “.gpx” (folglich GPS-Dateien genannt). |
| 1.1. | Einlesen | Programmstart | Wenn die Applikation gestartet wird, sollen alle GPS-Dateien im ersten Unterverzeichnis des beim Programmstart vom Nutzer gewählten Verzeichnis durch die Applikation eingelesen werden. |
| 1.1.1. | Einlesen | Programmstart -Nebenbedingung | Falls sich im gewählten Verzeichnis kein weiteres Unterverzeichnis befindet, soll die Applikation keine GPS-Dateien einlesen. |
| 1.2. | Einlesen | Verzeichniswechsel | Wenn der Nutzer das gewählte Verzeichnis wechseln möchte, soll die Applikation eine entsprechende Möglichkeit dafür anbieten. |
| 1.3. | Einlesen | Aktualisierung | Wenn der Nutzer das Aktualisieren der “GPS-Tracks” auswählt, soll die Applikation alle im aktuell gewählten Unterverzeichnis befindlichen GPS-Dateien einlesen. |
| 1.3.1. | Einlesen | Aktualisierung -Nebenbedingung | Falls sich im Unterverzeichnis eine Datei mit einem “GPS-Track”, welche schon in der Applikation vorhanden ist, befindet, dann soll die Applikation das Einlesen |

| | | | |
|---|---|---|---|
| | | | dieser Datei nicht durchführen. |
| 1.4. | Einlesen | zu parsende Informationen | Wenn eine Datei geparst wird, soll die Applikation folgende Informationen, falls vorhanden, über den darin enthaltenen “GPS-Track” in der Applikation speichern: Name des “GPS-Tracks”, Datum der Datenaufzeichnung, Startzeitpunkt der Datenaufzeichnung, zurückgelegte Distanz, benötigte Zeit, durchschnittliche Pace, durchschnittliche Geschwindigkeit, durschnittlicher Herzschlag pro Minute, maximaler Herzschlag pro Minute, positiver Höhenunterschied und die enthaltenen GPS-Datenpunkte. |
| 1.4.1. | Einlesen | zu parsende Informationen<br>- Nebenbedingung | Der Name des “GPS-Tracks” ist ident dem Namen der dazugehörigen Datei. |
| 1.4.2. | Einlesen | zu parsende Informationen<br>- Nebenbedingung | Wenn die GPS-Datenpunkte eingelesen werden, soll die Applikation ebenso, falls vorhanden, pro GPS-Datenpunkt eine fortlaufende Nummer innerhalb des “GPS-Tracks”, die zurückgelegte Distanz, die benötigte Zeit, die durchschnittliche Pace, die durchschnittliche Geschwindigkeit, der durschnittlicher Herzschlag pro Minute, der maximaler Herzschlag pro Minute und den positiver Höhenunterschied in der Applikation speichern. |
| 1.5. | Einlesen | Fehlermeldung | Falls ein Fehler während dem Parsen einer Datei auftritt, soll die Applikation das Einlesen dieser Datei abbrechen und mittels Fehlermeldung den Nutzer informieren. Weitere Dateien im gewählten Verzeichnis sollen jedoch weiter eingelesen werden. |
| 1.6. | Einlesen | Annahme | Annahme: Es werden nur Dateien im Format “.tcx” (Schemaversion 2) und “.gpx” (Schemaversion 1.1) akzeptiert. |
| 1.6.1. | Einlesen | Annahme<br>- Nebenbedingung | Jede Datei bildet einen einzulesenden GPS-Track. |
| | | | |
| 2. | Entfernen | Allgemeines zum Entfernen | Die Applikation soll das Entfernen von eingelesenen „GPS-Tracks“ ermöglichen. |

| | | | |
|---|---|---|---|
| 2.1. | Entfernen | Entfernung | Wenn der Nutzer das Aktualisieren der “GPS-Tracks” auswählt, soll die Applikation “GPS-Tracks”, welche keine zugehörige Datei im gewählten Unterverzeichnis aufweisen, entfernen. |
| 2.2. | Entfernen | Fehlermeldung | Falls ein Fehler beim Entfernen auftritt, soll die Applikation mittels einer Fehlermeldung den Nutzer darüber informieren. |
| | | | |
| 3. | Gruppierung | Allgemeines zur Gruppierung | Die Applikation soll das Gruppieren von „GPS-Tracks“ nach Zeiträumen ermöglichen. |
| 3.1. | Gruppierung | Gruppierungsargumente | Diese Gruppierung kann nach dem vollem Datum, wöchentlich (innerhalb von Jahr), monatlich (innerhalb von Jahr) oder jährlich geschehen. |
| 3.1.1. | Gruppierung | Gruppierungsargumente<br>-Nebenbedingung | Annahme: Die Gruppierungsargumente sind fixiert und können zur Laufzeit nicht erweitert werden. |
| 3.2. | Gruppierung | Gruppierung präsentieren | Wenn der Nutzer die Gruppierung auswählt, soll die Applikation die “GPS-Tracks” entsprechend dem Gruppierungsargument gruppieren und in der primären Listenansicht (siehe Punkt 4) als Gruppen präsentieren. |
| 3.2.1. | Gruppierung | Gruppierung präsentieren<br>- Nebenbedingung | Wenn der Nutzer ein Gruppierungsargument auswählt, soll die Applikation folgende Informationen pro Gruppe in der primären Listenansicht anzeigen: Zeitraum der Gruppierung, Anzahl der Elemente in der Gruppe, kombinierte zurückgelegte Distanz, kombinierte benötigte Zeit, durchschnittliche Pace, durchschnittliche Geschwindigkeit, durchschnittlicher Herzschlag pro Minute, maximaler Herzschlag pro Minute und kombinierter positiver Höhenunterschied. |
| 3.2.2. | Gruppierung | Gruppierung präsentieren<br>- Nebenbedingung | Wenn der Nutzer eine Gruppe in der primären Listenansicht auswählt, soll die Applikation alle darin enthaltenen “GPS-Tracks” dieser Gruppe in der sekundären Listenansicht anzeigen (siehe Punkt 5). Die anzuzeigenden Informationen pro “GPS-Track” sind dieselben wie bei Punkt 4.1. |

| | | | |
|---|---|---|---|
| 3.3. | Gruppierung | Fehlermeldung | Falls ein Fehler auftritt, soll die Applikation die primäre Listenansicht beziehungsweise sekundäre Listenansicht leer angezeigt werden. |
| | | | |
| 4. | Primäre Listenansicht | Allgemeines zur primären Listenansicht | Die Applikation soll eine primäre Listenansicht unterstützen. |
| 4.1. | Primäre Listenansicht | “GPS-Tracks” präsentieren | Während die primäre Listenansicht der „GPS-Tracks“ aktiv ist, soll die Applikation pro “GPS-Track”, falls vorhanden, den Namen, das Datum der Datenaufzeichnung, den Startzeitpunkt der Datenaufzeichnung, die zurückgelegte Distanz, die benötigte Zeitdauer, die durchschnittliche Pace, die durchschnittliche Geschwindigkeit, der durchschnittliche Herzschlag pro Minute, der maximale Herzschlag pro Minute und den positive Höhenunterschied anzeigen. |
| 4.2. | Primäre Listenansicht | Attributskombination | Unter der primären Listenansicht sollen alle “GPS-Tracks” beziehungsweise die Gruppierungen hinsichtlich ihrer Attribute (siehe Punkt 4.1 bzw. 3.2) kombiniert werden und entsprechend eine Summe oder ein Durchschnitt pro Attribut angezeigt werden. |
| 4.2.1. | Primäre Listenansicht | Attributskombination - Nebenbedingung | Name, Datum der Datenaufzeichnung und Startzeitpunkt der Datenaufzeichnung sind nicht zu kombinieren. |
| | | | |
| 5. | Sekundäre Listenansicht | Allgemeins zur sekundären Listenansicht | Die Applikation soll eine sekundäre Listenansicht unterstützen. |
| 5.1. | Sekundäre Listenansicht | Segmentierung | Wenn der Nutzer einen einzelnen “GPS-Track” in der primären Listenansicht auswählt, sollen die in diesem “GPS-Track” enthaltenen einzelnen aufgezeichneten GPS-Datenpunkte laut einem Segmentierungsargument von der Applikation gruppiert und dargestellt werden. |
| 5.1.1. | Sekundäre Listenansicht | Segmentierung - Nebenbedingung | Die Segmentierung kann dabei mittels 1 Meter, 10 Meter, 100 Meter, 400 Meter, 500 Meter, 1000 Meter, 5000 Meter, 10000 Meter, ein Viertel eines Marathons, ein |

| | | | |
|---|---|---|---|
| | | | halber Marathon und pro gespeichertem GPS-Datenpunkt erfolgen. Bei Auswahl einer Distanz sollen die GPS-Datenpunkte hinsichtlich ihrer zurückgelegten Distanz addiert werden bis das gewünschte Segmentierungsargument am nächsten erreicht wird. Diese GPS-Datenpunkte bilden gemeinsam ein Segment und es sollen alle Informationen pro Segment, wie bei Punkt 5.1.3 erläutert, errechnet werden. Dies wird fortgesetzt bis alle GPS-Datenpunkte einem Segment zugewiesen sind. Während die Applikation ausgeführt wird, ist ein Segmentierungsargument immer aktiv. Das voreingestellte Argument bei Applikationsstart ist GPS-Datenpunkt. |
| 5.1.2. | Sekundäre Listenansicht | Segmentierung - Nebenbedingung | Annahme: Die möglichen Segmentierungsargumente sind zur Laufzeit fixiert und können nicht erweitert werden. |
| 5.1.3. | Sekundäre Listenansicht | Segmentierung - Nebenbedingung | Wenn der Nutzer einen einzelnen “GPS-Tracks” in der primären Listenansicht auswählt, sollen folgende Informationen, falls vorhanden, auf der sekundären Listenansicht pro Segment präsentiert werden: Nummer des Segmentes, kombinierte zurückgelegte Distanz, kombinierte benötigte Zeit, die durchschnittliche Pace, die durchschnittliche Geschwindigkeit, der durchschnittliche Herzschlag pro Minute, der maximale Herzschlag pro Minute und der positive Höhenunterschied. |
| 5.1.4. | Sekundäre Listenansicht | Segmentierung - Nebenbedingung | Wenn innerhalb der primären Listenansicht kein Eintrag gewählt worden ist, soll die Applikation die sekundäre Listenansicht als leer anzeigen. |
| 5.2. | Sekundäre Listenansicht | Fehlermeldung | Falls ein Fehler auftritt, soll die Applikation die sekundäre Listenansicht als leer anzeigen. |
| | | | |
| 6. | Filter | Allgemeines zur Filterung | Die Applikation ermöglicht das Filtern der “GPS-Tracks” nach einer vom Nutzer kreierten Filterliste. |
| 6.1. | Filter | Filter-Einträge | Die möglichen Filter-Einträge basieren dabei auf der Verzeichnisstruktur des aktuell aktiv gewählten Verzeichnis. |

| | | | |
|---|---|---|---|
| 6.1.1. | Filter | Filter-Einträge<br>- Nebenbedingung | Ein Filter-Eintrag entspricht einem Unterverzeichnis aus dem aktuell aktiv gewählten Verzeichnis. |
| 6.2. | Filter | Auswahl eines Filter-Eintrags | Wenn der Nutzer einen neuen Filter-Eintrag auswählt, soll die Applikation alle aktuellen “GPS-Tracks” durch jene in dem entsprechendem Verzeichnis ersetzen. |
| 6.3. | Filter | Voreinstellung | Während die Applikation ausgeführt wird, ist ein Filter-Eintrag immer aktiv. Der voreingestellte Filter-Eintrag bei Applikationsstart ist, falls vorhanden, das erste Verzeichnis. |
| | | | |
| 7. | Balkendiagramm | Allgemeines zum Balkendiagramm | Die Applikation soll ein Balkendiagramm zum Vergleich der “GPS-Tracks” unterstützen. |
| 7.1. | Balkendiagramm | y-Achse-Argumente | Die Argumente auf der y-Achse können zurückgelegte Distanz, benötigte Zeit, durchschnittliche Pace, durchschnittliche Geschwindigkeit, durschnittlicher Herzschlag pro Minute, maximaler Herzschlag pro Minute und positiver Höhenunterschied sein. |
| 7.1.1. | Balkendiagramm | y-Achse-Argumente<br>- Nebenbedingung | Wenn der Nutzer ein neues Argument für die y-Achse auswählt, soll die Applikation das Balkendiagramm entsprechend dem gewählten Argument anpassen. |
| 7.1.2. | Balkendiagramm | y-Achse-Argumente<br>- Nebenbedingung | Während die Applikation ausgeführt wird, ist ein y-Achse-Argument immer aktiv. Das voreingestellt Argument ist zurückgelegte Distanz. |
| 7.1.3. | Balkendiagramm | y-Achse-Argumente<br>- Nebenbedingung | Annahme: Die y-Achse-Argumente sind fixiert und können nicht zur Laufzeit erweitert werden. |
| 7.2. | Balkendiagramm | Balkendiagramm präsentieren | Während die Applikation ausgeführt wird, soll ein Balkendiagramm angezeigt werden, welches sich der Auswahl des Nutzers zur Laufzeit anpasst. |
| 7.2.1. | Balkendiagramm | Balkendiagramm präsentieren<br>- Nebenbedingung | Während sich die Applikation in der primären Listenansicht ohne Gruppierung befindet, sollen alle “GPS-Tracks”, die sich in der Listenansicht befinden, laut Argument der y-Achse dargestellt werden. Die Namen |

| | | | |
|---|---|---|---|
| | | | der “GPS-Tracks” bilden die Inhalte der x-Achse. |
| 7.2.2. | Balkendiagramm | Balkendiagramm präsentieren<br>- Nebenbedingung | Während sich die Applikation in der primären Listenansicht ohne Gruppierung aber mit ausgewähltem “GPS-Track” befindet, sollen alle Segmente dieses “GPS-Tracks” (siehe Punkt 5.1) laut Argument der y-Achse dargestellt werden. Die Nummer der Segmente bilden die Inhalte der x-Achse. |
| 7.2.3. | Balkendiagramm | Balkendiagramm präsentieren<br>- Nebenbedingung | Während sich die Applikation in der primären Listenansicht mit Gruppierung befindet, sollen alle Gruppierungen laut Argument der y-Achse dargestellt werden. Der Zeitraum der Gruppierungen bilden die Inhalte der x-Achse. |
| 7.2.4. | Balkendiagramm | Balkendiagramm präsentieren<br>- Nebenbedingung | Während sich die Applikation in der primären Listenansicht mit Gruppierung und ausgewählter Gruppe befindet, sollen die in der Gruppe befindlichen “GPS-Tracks” laut Argument der y-Achse dargestellt werden. Die Namen der “GPS-Tracks” in der Gruppe bilden die Inhalte der x-Achse. |
| 7.3. | Balkendiagramm | Fehlermeldung | Falls ein Fehler auftritt, soll die Applikation das Balkendiagramm leer anzeigen. |
| | | | |
| 8. | Vergleich | Allgemeines zum Vergleich | Die Applikation soll einen jährlichen Vergleich der “GPS-Tracks” unterstützen. |
| 8.1. | Vergleich | Vergleichsargumente | Wenn der Nutzer die entsprechenden Jahre auswählt, sollen diese Jahre anhand des aktuellen Gruppierungsargument auf dem Balkendiagramm mit dem aktiven y-Achse-Argument (siehe Punkt 7.1) verglichen werden. |
| 8.1.1. | Vergleich | Vergleichsargumente<br>- Nebenbedingung | Falls kein Gruppierungsargument aktiv ist, soll die Applikation den Vergleich mittels Gruppierungsargument “Monat” durchführen. |
| 8.1.2. | Vergleich | Vergleichsargumente<br>- Nebenbedingung | Die möglichen wählbaren Jahre sind dabei abhängig von den eingelesenen “GPS-Tracks”. |

| | | | |
|---|---|---|---|
| 9. | Suche | Allgemeines zur Suche | Die Applikation soll die Suche nach den Namen der “GPS-Tracks” ermöglichen. |
| 9.1. | Suche | Suchfunktionsweise | Wenn der Nutzer in das Suchfeld einen Suchbegriff eingibt, soll die Applikation die zu der Zeit aktiv angezeigte Liste in der primären Listenansicht (siehe Punkt 4) nach den entsprechenden Namen der “GPS-Tracks” filtern. Es sollen nur jene “GPS-Tracks” angezeigt werden, welche diesem Suchbegriff entsprechen. |
| 9.1.1. | Suche | Suchfunktionsweise - Nebenbedingung | Falls ein Gruppierungsargument bei der Suche aktiv ist, soll die Applikation nur jene Gruppen präsentieren, welche “GPS-Tracks”, die dem Suchbegriff entsprechen, innehaben. |
| | | | |
| 10. | Schließen | Allgemeines zum Schließen | Die Applikation soll das Schließen der Applikation ermöglichen. |
| 10.1. | Schließen | Fehlermeldung | Wenn der Nutzer die Applikation schließt, soll die Applikation alle offenen Fenster schließen. |

# Nicht-Funktionale Anforderungen

Zur Klärung der Nicht-Funktionalen Anforderungen orientieren wir uns an den ISO-9126 Standard.

## Wartbarkeit

Die Applikation soll möglichst wartbar gestaltet werden. Dafür sollen entsprechende “Best-Practices” der Programmiersprache Java aber auch allgemeinere “Best-Practices” der objekt-orientierten Programmierung eingehalten werden.

Hinsichtlich Stabilität soll pro 100 Stunden Betriebszeit höchstens ein Absturz der Applikation vorfallen.

Zur Sicherstellung der Wartbarkeit sollen ebenso nach einer Analyse des Programms “SonarQube” weder “blocker”, “critical” oder “major” Issues erkannt werden. Dadurch wird ebenso eine gute Analysierbarkeit und Testbarkeit sichergestellt.

## Benutzbarkeit

Die Applikation soll klar und verständlich für den Nutzer sein. Ein Nutzer sollte die Applikation während der erstmaligen Nutzung vollstens bedienen können ohne die Benutzerdokumentation zu konsultieren. Die Nutzung muss daher möglichst intuitiv sein und für den Nutzer einem bereits bekannten Nutzungsstil entsprechen.

Die Benutzeroberfläche muss daher möglichst modern, ästhetisch aber auch funktionell sein.

## Effizienz

Die Applikation soll möglichst ressourcenarm in der Ausführung sein. Es sollen während der Ausführung bei einer Anzahl von 250 eingelesenen “GPS-Tracks” maximal 250 Megabyte Arbeitsspeicher verwendet werden.

Beim Start der Applikation soll das Einlesen und Anzeigen von “GPS-Tracks” mit einer kombinierten Speichergröße von 75 Megabyte höchstens fünf Sekunde benötigen.

Das Ändern eines Gruppierungsarguments oder die Nutzung der Vergleichsfunktion und der damit einhergehenden Umstellung der graphischen Oberfläche darf bei 500 “GPS-Tracks” höchstens eine halbe Sekunde benötigen.

Das Ändern des Filter-Eintrags oder des gewählten Verzeichnisses und der damit einhergehenden Umstellung der graphischen Oberfläche darf bei “GPS-Tracks” mit einer kombinierten Speichergröße von 75 Megabyte höchstens fünf Sekunde benötigen.

Das Aktualisieren der vorhandenen “GPS-Tracks” während der Laufzeit und der damit einhergehenden Umstellung der graphischen Oberfläche darf bei “GPS-Tracks” mit einer kombinierten Speichergröße von 75 Megabyte höchstens fünf Sekunde benötigen.

Jegliche andere Interaktionen sollen innerhalb von höchstens einer halben Sekunde abgeschlossen sein.

## Funktionalität

Die Applikation soll nur jene Funktionen anbieten, die in ihrem Einsatzkontext auch Sinn ergeben. Diese Funktionen müssen richtig (ergo den Anforderungen entsprechend) implementiert werden.

Die Applikation interagiert mit keinen anderen Anwendungssystem und muss dafür entsprechend keine Unterstützung anbieten.

Die Applikation interagiert nicht mit anderen Diensten und hat ebenso keine direkten Netzwerkverbindungen. Jegliche Daten werden ebenso nur während der Ausführung in der Applikation gehalten und werden (außerhalb des Aufgabengebietes der Applikation) persistent in durch vom Betriebssystem verwaltete Dateien gehalten. Der Sicherheitsaspekt ist daher zu vernachlässigen.

Die Applikation soll die Daten korrekt verarbeiten. Die Gruppierung, Suche und Filterung von “GPS-Tracks” ist korrekt durchzuführen. Beim Einlesen der GPS-Dateien sollen ebenso alle notwendigen Informationen korrekt und vollständig ausgelesen werden.

## Übertragbarkeit

Durch die Nutzung von Java als Programmiersprache ist eine hohe Übertragbarkeit von Beginn an gegeben. Die Applikation soll auf “java-fähigen” (ab Version 11) Computern ausführbar sein. Die Interaktion mit dem Nutzer findet über Maus und Tastatur statt. Eine Installation der Applikation ist nicht notwendig, da die Applikation mittels Java sofort ausgeführt werden kann.

Eine möglichst gute Koexistenz mit anderen Programmen soll gegeben sein. Etwaige Ressourcennutzungen, wie zum Beispiel den GPS-Dateien beim Einlesen, sollen nach Abschluss oder einer Fehlermeldung wieder freigegeben werden.

## Zuverlässigkeit

Die Applikation soll Fehlermeldungen, welche innerhalb des Adressierungsbereich des Entwicklers liegen (Fehler in der “Java Virtual Machine” als Gegenbeispiel), behandeln und Funktionen bestmöglich fortsetzen können.

Besonders beim Einlesen von GPS-Dateien darf es aufgrund von zum Beispiel einer fehlerhaften Datei zu keinem Absturz kommen. Nach einem Absturz der Applikation soll nach erneutem Start und erneutem Einlesen der GPS-Dateien die Nutzung wieder fortgefahren werden können.

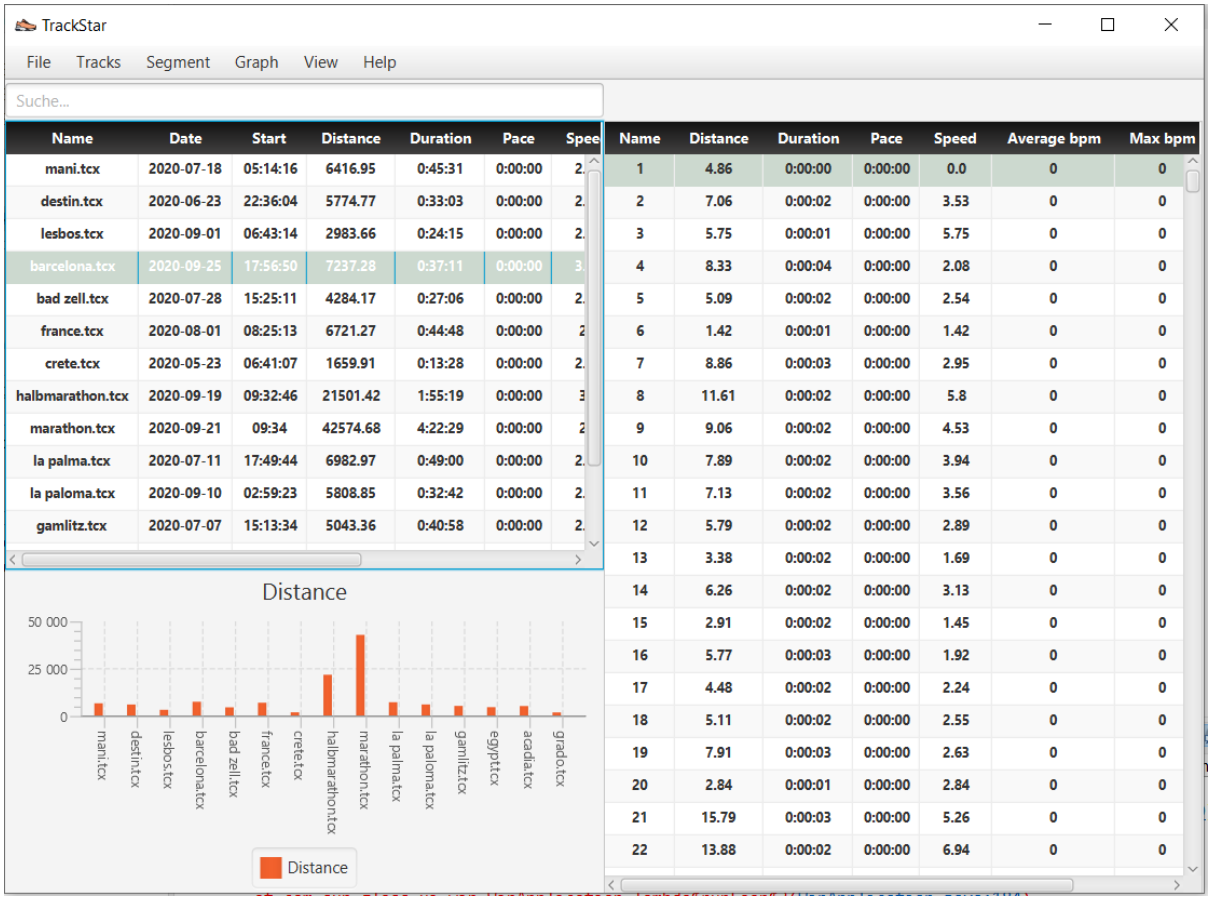
# Aufbau der Applikation

Die Applikation besteht grundsätzlich aus einem Fenster. Dieses Fenster enthält im oberen Bereich eine Menüleiste, welche die möglichen Interaktionsmöglichkeiten für den Nutzer zum Einstellen der gewünschten Argumente und Sichten enthält. Zur Darstellung der Inhalte wird das Fenster in drei Bereiche aufgeteilt.

Wie auf der Abbildung ersichtlich, befindet sich links oben eine primäre Listenansicht zum Darstellen der einzelnen “GPS-Tracks” oder der Gruppierungen. Diese primäre Listenansicht ist dabei interaktiv und aktiviert bei Auswahl die sekundäre Listenansicht (auf der Abbildung rechts anzufinden).

Diese sekundäre Listenansicht stellt je nach gewählter Ansicht auf der primären Listenansicht entweder “GPS-Tracks” oder Segmente dar.

Je nach gewählter und aktiver Argumente wird unten links in dem Fenster ein entsprechendes Balkendiagramm angezeigt.

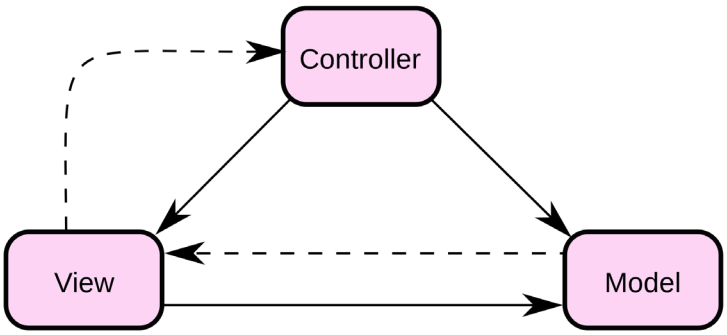


# Architektur

Zur Umsetzung dieser Applikation wird Java Version 11 verwendet. Zur Realisierung der graphischen Benutzeroberfläche wird das Framework “JavaFX” in der Version 18 verwendet. Die Applikation wird mittels einem “MVC-Pattern” realisiert, da “JavaFX” fundamentall dafür strukturiert und ebenso für die Aufgabenstellung passend ist. Das “MVC-Pattern” besteht aus einem Modell, einem oder mehreren “Views” und der Anzahl der “Views” entsprechenden “Controllers”.

Das Modell enthält die für die Applikation notwendigen Daten und die entsprechende Geschäftslogik. Das sind zum Beispiel Methoden um Elemente hinzuzufügen oder sie auch wieder zu entfernen. In unserem Projekt verwenden wir das Modell um alle “GPS-Tracks” und weitere notwendige Informationen zu verwalten. Zum Interagieren mit dem Modell und dem Liefern von Informationen an die “Views” wird ein “Controller” verwendet. Dieser definiert welche Aktion beim Drücken von zum Beispiel einem Knopf auf der “View” ausgelöst werden soll. Ebenso bestimmt der “Controller”, welche Daten aus dem Modell auf der “View” zu sehen sind. Unsere Applikation besteht aus einem “Controller”. Dieser “Controller” übernimmt die Steuerung der Listenansichten, des Diagramms und der allgemeinen Interaktion mit dem Nutzer. Die “View” übernimmt die Darstellung der Interaktionsmöglichkeiten und der Daten. Die “View” kommuniziert dazu mit dem “Controller”. In unserem Projekt interagiert die “View” mit dem “Controller” um die Daten für die Darstellung der Listenansichten und dem Balkendiagramm zu beschaffen.

Durch die Auswahl eines “MVC-Pattern” sind die Komponenten der Applikation eine Main-Klasse zum Start der Applikation, ein Modell, mehrere “Views” und entsprechende “Controllers”. Der “TrackManagerController” ist dabei der Haupt-Controller, welcher die Interaktion mit dem Hauptfenster übernimmt.




Eine genauere Auflistung der Komponenten der Applikation ist dem UML-Klassendiagramm zu entnehmen.

Die graphische Benutzeroberfläche wird, wie schon angesprochen, mittels “JavaFX”, aber auch mittels “FXML” umgesetzt. Der Vorteil durch den Einsatz von “FXML” ist die Trennung zwischen der graphischen Komponente (“View”) und der Logik, welche mit der Grafik interagiert (“Controller”). Dies ermöglicht es die graphische Benutzeroberfläche anzupassen ohne Änderungen an der Logik vornehmen zu müssen.