

**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ  
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

**Тема: Приложение за анализ и контрол на тренировъчния процес  
в академичното гребане.**

Дипломант:  
Божидар Захов

Научен ръководител:  
Пламен Танов

СОФИЯ  
2017



## УВОД

Академичното Гребане е колективен воден спорт, произхождащ от Древен Египет. Той е един от най-старите Олимпийски спортове и е популярен из целия свят. Целта му е да се постигне максимална скорост на лодката, като ускоряването се достига и поддържа с човешка сила, предадена чрез греблата, като движението на гребца е подпомогнато от плъзгането на слайда. Спортистът е разположен с гръб по посоката на движението. В академичното гребане има два вида - скулово и разпашно. При скуловото гребане състезателите държат с всяка ръка по едно гребло, като за него е необходима по-добра техническа подготовка. При разпашното гребане всеки от състезателите държи по едно гребло, като при него силата на гребане е от по-голямо значение.

Спортът академично гребане може да се практикува както за отдих, така и състезателно. При състезанията, лодките се надпреварват да достигнат максимално бързо до финалната линия, която е на дистанция от две хиляди метра. Обикновено трасето е разделено на шест коридора, във всеки коридор е разположена по една лодка, в която в зависимост от вида, може да участва и управляващ лодката и задаващ темпото състезател наречен кормчия. Има различни видове лодки - състезателни, обучителни и морски, като всеки вид могат да бъдат:

- едноместни - скифове (1х);
- двуместни, които могат да са:
  - скулови (2х);
  - разпашни:
    - с кормчия (2+);
    - и без кормчия (2-);
- четири местни, които могат да са:
  - скулови (4х), обикновено са без кормчия;
  - разпашни:

- с кормчия (4+);
- без кормчия (4-);
- осем местни, те са само разпашни с кормчия (8+);
- двадесет и четири местни, които са скулови и се срещат рядко, тъй като няма такава Олимпийска дисциплина.

При състезателното гребане е много важно да се съчетава перфектна техника, отлична физическа подготовка и добре разработена стратегия при самото състезание, част от която е правилното разпределение на силите на спортиста, което е и целта на това приложение.

Техниката е най-трудна за усъвършенстване част от състезателния процес. Екипажът трябва да поддържа лодката балансирана, спазвайки спецификата на движението и в същото време да прилага максимално разумната сила за всеки етап от състезанието, без да се влияе от външните фактори, като време, вълнение и други. Затова, всеки един състезател и треньорът му биха желали да имат на разположение спомагателни средства, чрез които да подобрят представянето и чрез които да разбират ефективността на вложеният труд в реално време, както и разбор на дейността след края на тренировката.

Целите на този проект са да се подпомогнат спортистите и техните треньори с качествена и важна информация за текущото състояние и способности на всеки един състезател. Да се проследява във времето развитието на състезателя, както и при разбора на тренировките да се улесни намирането на грешките допускани от него, за да се постигне желания резултат.

# **1. ПЪРВА ГЛАВА.**

## **ПРЕГЛЕД НА СЪЩЕСТВУВАЩИ ПОДОБНИ ПРОДУКТИ**

### **1.1. Проблематика**

Всеки спортист знае, че е трудно проследяването на ефективността на тренировката и следенето на допуснатите грешки. Този процес може да бъде подпомогнат от приложенията „помощници“, на състезателите и техните треньори, в спорта академично гребане. Целта на тези приложения е да помагат на използвателите им в различните видове дейности и изчисления. Така състезателите да бъдат съсредоточени в тренировъчния процес, а след него, техните треньори да имат данни подпомагащи разбора на тренировката. По този начин се цели подобряването и ефективността на цялостния процес.

### **1.2. Съществуващи продукти**

#### **1.2.1. Endomondo**

– уебсайт с мобилно приложение за спортисти, предлагащо множество функции, то е едно от най-известните и продължава да набира широка популярност. Разполага с привлекателен дизайн и може да бъде свалено свободно за Android и iPhone. Приложението предлага следене на калории, следене на тренировъчен режим, отбелязване на изминати километри и други параметри.

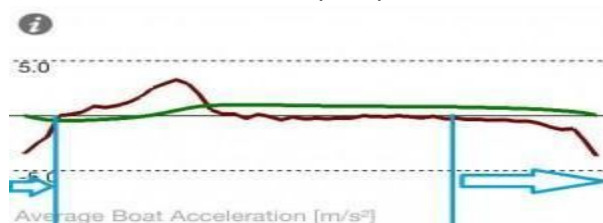
Фиг. (1.1)

#### **1.2.2. RowingInMotion (RiM)**

– специализирано Андроид приложение за академично гребане, което следи тренировъчния процес. Разполага със сайт, където всеки желаещ може да се научи на някои

тънкости в гребането. То предлага множество специализирани функции, чрез които спортистът и/или неговият треньор, в реално време, или след завършването, може да свери представянето и ефективността на тренировката.

Фиг. (1.2)



Фиг. (1.3)

### 1.3. Обобщение на съществуващите продукти:

#### 1.3.1. Ендомондо

То е безплатно приложение за спортни дейности, но то не е специализирано за спорта академично гребане и може да следи само някои основни функции, за конкретния спорт.

#### 1.3.2. RowingInMotion

То е много добро приложение за анализ в спорта академично гребане, но то е платено, поради което е по-трудно достъпно.

## **2. ВТОРА ГЛАВА.**

### **ПРОЕКТИРАНЕ НА СТРУКТУРАТА НА АНДРОИД БАЗИРАНО МОБИЛНО ПРИЛОЖЕНИЕ ЗА АНАЛИЗ И КОНТРОЛ НА ТРЕНИРОВЪЧНИЯ ПРОЦЕС**

#### **2.1. ИЗБОР НА ИЗПОЛЗВАНИТЕ ТЕХНОЛОГИИ**



Фиг.(2.1)

##### **2.1.1.Избиране на операционна система и програмен език**

За разработването на приложението е използван “Android Software Development Kit” (Android SDK), който осигурява всички необходими инструменти за това като компилатор, дебъгер, библиотеки, емулатор, както и документация.

##### **2.1.1.1. Избор на език**

За език за програмиране беше избран Java, поради:

- o Java е обектно ориентиран език за програмиране. Осигурява разделяне на приложението на самостоятелни модули и по този начин се постига по-голяма гъвкавост, която е изключително важна при бъдещето развитие на приложенията разработвани чрез нея.
- o Освен това се осигурява капсулация на данните и

вътрешното устройство(клас), като достъпа до тези данни се определя от специално дефинирани интерфейси. По този начин се осигурява независимо развитие на всеки компонент от програмната система, с минимална зависимост от останалите.

- **Плюсовете на Java са:**

- Лесната преносимост между различните платформи - Софтуерни или хардуерни, тъй като веднъж написана и компилирана, една Java програма може да бъде стартирана на компютри независимо от архитектурата или от операционната им система;
- Допълнителните действия, извършвани от виртуалната машина като освобождаване на паметта от класове, които не се използват (Garbage collector);
- Висока степен на сигурност поради факта, че програмистите не работят директно с паметта на устройството и други предимства.

Приложението е написано на програмния език “Java” и се използва специална виртуална машина - “Android Runtime (ART)”.

### **2.1.2.Компонентите на едно Android приложение са:**

- “Activity” - репрезентира презентационния слой на приложението;
- “Content Provider” - предоставя структуриран интерфейс към данните на приложението, чрез който други приложения могат да достъпват до тази информация. Данните могат да бъдат съхранявани по различен начин, като например във файловата система или в “SQLite” база данни;
- “Broadcast Receiver” - този вид компонент получава и отговаря на системни съобщения и “Intent” - и. Той ще бъде уведомен от Android система, ако възникне определено събитие, като промяна в нивото на батерията, смяна на езика и други;
- “Services” - това са услуги, които изпълняват задачи на заден фон без да предоставят потребителски интерфейс. Те могат да



уведомяват потребителите чрез системата за известия (“notifications”) в Android.

### **2.1.3. Други използвани термини:**

- Изгледи (“Views”) - представляват различни части от потребителския интерфейс като бутони, текстови полета и други. Базовият клас за всички изгледи е “android.view.View”;
- “Intent” - това са асинхронни съобщения, които изискват някаква функционалност от други компоненти като “Services” или “Activities”. Приложението може да извика компонента директно (“explicit Intent”) или да използва “Broadcast receivers” за получаването на “Intent” - и (“implicit Intent”).

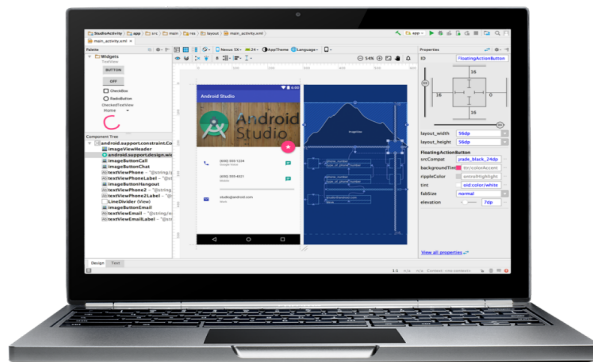
### **2.1.4. Избор на средата за разработка(IDE)**

Изборът е приложението да се напише чрез Android Studio, тъй като тя вече е единствената официална платформа за разработка на Android приложения. Алтернативата беше Eclipse с приставката ADT(Android Developer Tools), но Google вече не поддържа ADT и съответно очакват работата си по “Android Studio”, което е базирано на средата за разработка за „Java“ на „JetBrains, IntelliJ IDEA“. То също също така се счита за официалното такова за Android разработване.

Някои от функционалностите, с които „Android Studio“ разполага са:

- Gradle build системата - конфигурация за “създаване” на изпълнимия файл (.apk);
- Редактор;
- Интеграция на Version Control System (Git);
- Създаване на виртуални устройства с всякакви размери и форми (телевизори, телефони, таблети и др.);
- Шаблони с код;
- Графичен интерфейс за .xml layout файловете;
- Lint - анализатор на кода за потенциални бъгове и възможни оптимизации;
- ProGuard - намалява размера на кода, като премахва

неизползваните части от него и преименува някои класове.



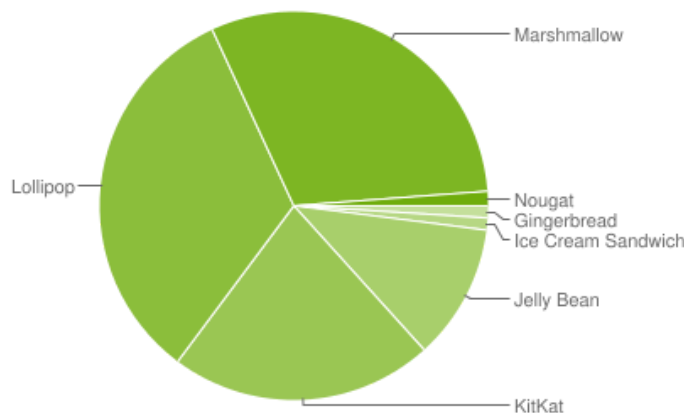
Фиг. (2.2)

### 2.1.5. Избиране на версия на Android

Последната версия на Android е 7 - Nougat. Новите версии на Андроид имат в себе си повече функционалност, която може да бъде използвана от програмиста и показана на потребителя, но само ако неговата версия е инсталирана на операционната система е същата или по-висока, от колкото функционалността на приложението:

- Например: runtime permissions навлизат в API 22 (Marshmallow) и затова писането на приложение, което върви с функционалност от последната Android версия, няма да може да бъде използвана от много потребители, защото тяхната версия ще е по-ниска от тази на приложението. Затова разработването на програми за Nougat в момента няма добра опция, освен евентуално, ако не е предвидено изчакване за популяризирането и.

Тъй като от приложението се изисква да е на разположение на възможно най-много потребители, от очакваната целева аудитория и да е съобразено с минималната необходима функционалност, то на разработчика се налага да балансира в избора си между достъпност и функционалност. Например ако се използва API 1, то приложението ще е на разположение на всички потребители, но няма да разполага с достатъчно вградени функции, с които да реализира целта на приложението. За разработката на настоящата дипломна работа беше избрано API 23 (Marshmallow), тъй като повече 30% от потребителите използват това или по-високо API.



Фиг. (2.3)

Тази графика е създадена с данните от седем дневния период между 1 Януари и 6 Февруари 2017-та година. Всички версии, използвани по-малко от 0.1 процента от всички потребители, не са включени в горната диаграма. [\[2\]](#)

#### 2.1.6. Избор на зависимости (dependencies) на проекта

В разработката на софтуер, не е необходимо всичко да се създава от разработчика. Повечето от функционалността, която трябва да бъде използвана, вече е била създадена и употребена в други проекти. Използването на популярни библиотеки написани от други специалисти, спестява излишна работа на програмиста, чрез което се освобождава време за разработката на конкретната функционалност на новия продукт.

Друго предимство за използването им е, че начинаещ в създаването на този тип функционалност, често допуска грешки, докато при библиотеките, честа практика е обновяването и поддръжката им. Поради това с времето се заличават грешките и бъговете допуснати в кода, подобрява се бързодействието, развива се библиотеката с добавянето на нова функционалност и други.

В приложението са използвани външни библиотеки. За по-разбираемо описание, нека ги разделим на два вида:

- официални зависимости, невлизащи в SDK-а („com.android.support“);
- зависимости от трети лица, пуснати с лиценз за свободно използване;

#### **2.1.6.1. Официални зависимости:**

- 'com.android.support:appcompat-v7:24.0.0' – предлага класове съвместими и поддържащи по-стари версии на Android;
- 'com.android.support:recyclerview-v7:24.0.0' – предлага оптимизиран контейнер подобен на класа „ListView“ от Android SDK-а, но с тази разлика, че в него е имплементиран „viewholder pattern“, което води до по-добро представяне при голям обем от данни;
- 'com.android.support:design:24.0.0' – помощна библиотека за дизайн, съдържаща основните практики за потребителски интерфейс в Android.

#### **2.1.6.2. Зависимости на трети лица**

Външните библиотеки, използвани в този проект са:

##### **2.1.6.2.1. Firebase Authentication**

Повечето приложения трябва да знаят самоличността на даден потребител. Идентифицирайки го, това им позволява да запазват данни в облака и да осигуряват персонализирана работа във всички устройства на потребителя по всяко време и място.

Firebase Authentication осигурява Backend услуги, лесни за употреба SDKs и готови библиотеки за удостоверяване на потребителите в приложенията, които ги използват. Той поддържа стандартни методи за удостоверяване, като регистрация с имейл адрес и парола, както и популярни идентификационни платформи, като Google, Facebook, Twitter и GitHub.

Firebase Authentication се интегрира тясно с други Firebase услуги, спазвайки стандарти като OAuth 2.0 и OpenID Connect, така че да може лесно да се нагажда според нуждите на използвателя и.

##### **2.1.6.2.2. Firebase Realtime DataBase**

Firebase Database е NoSQL, база данни, която ви позволява да съхранявате и синхронизирате данни за всички клиенти в облака. Това

става по лесен и удобен начин, в реално време.

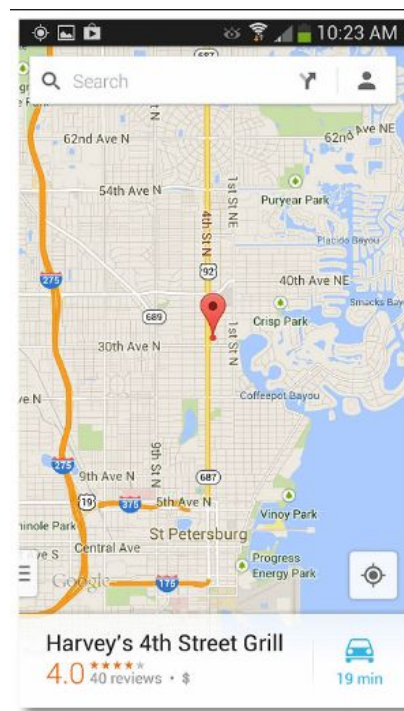
Данните се синхронизират между всички клиенти и остават на разположение на устройството, когато приложението загуби връзка с интернет. Освен това те се съхраняват, като JSON формат.

Всички клиенти споделят една Realtime Database инстанция и автоматично получават актуализации с най-новите данни, независимо от това от коя платформа са ( Android, IOS или JavaScript).

#### 2.1.6.2.3. Google Maps

Това API на Google Maps позволява вграждането на Google карти (Maps) към мобилни приложения на външни разработчици, с помощта на обикновен Java интерфейс. Проектирани са да работят на мобилни устройства, традиционни приложения за компютър, както и за интернет страници.

В андроид с тяхна помощ разработчиците разполагат със специален изглед (View) - “MapView”, който може да бъде вграден лесно и удобно в приложенията, имащи карта, разполагаща с всичките картографирані местоположения на Google, като сателитни снимки, пътища, информация за близки обекти и други. Този интерфейс разполага с много вградени функции за локализация и геокодиране, които подпомагат използвателите му.

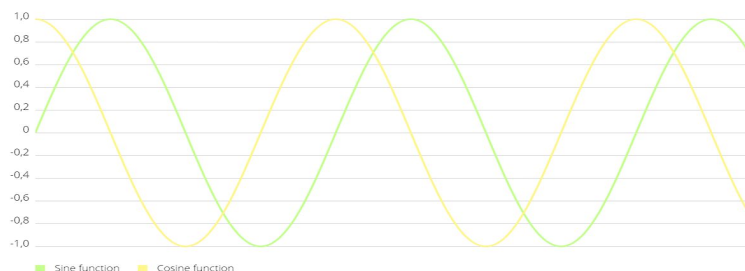


Фиг.(2.4)

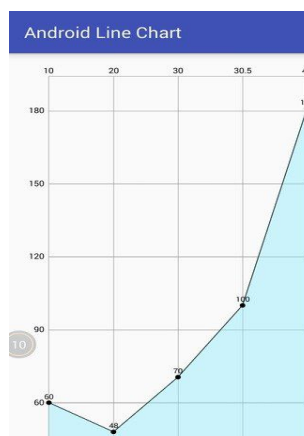
#### 2.1.6.2.4. MPAndroidChart

MPAndroidChart е мощна и удобна библиотека за Android, чрез която може да се създават осем вида различни графики. С тях е възможно да се изобразява графично информацията, с цел по-голяма прегледност и по-лесно възприемане и разбиране на числовата

информация от потребителя.



Фиг. (2.5)



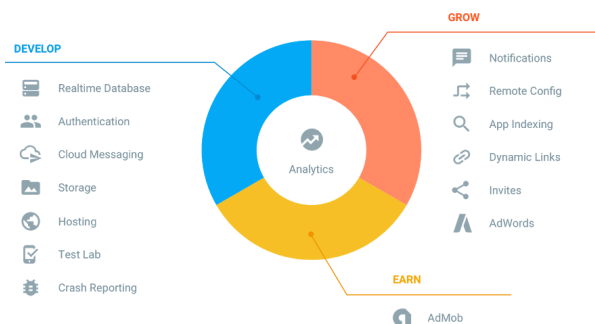
Фиг. (2.6)

#### 2.1.6.2.5. Избор на Бекенд(BackEnd) – Firebase Database

Изборът на бекенд не е лесен. Ваас бекенд като услуга (backend as a service) ви позволява да си осигурите база данни, която дава разрешение на потребителите да добавят обекти в облачните услуги.

В миналото или при нужда от специфична употреба, разработчиците са създавали сървърна част, като са използвали технологии като Ruby и PHP. Това често се оказва трудоемка и времеемка задача, която изисква ресурси, както и специални умения за създаване на продукт, който работи ефективно.

Изборът на бекенд зависи от лични предпочитания и какви са целите на приложението. В този проект е избрана Firebase Database поради удобния, лесен и бърз достъп до данните в облачните услуги.

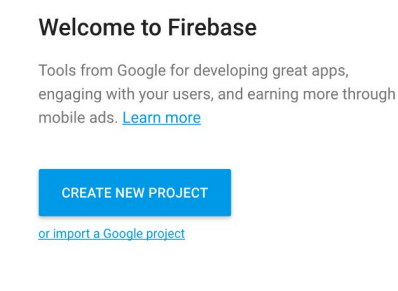


Фигура.2.7.

#### 2.1.6.2.5.1. Подготовка за свързване

Първоначално трябва да се създаде профил в сайта на Firebase (<https://firebase.google.com/>). Може да се използва и вече съществуващ Гоогъл акаунт (Google account).

След това трябва да се създаде нов проект чрез бутона “Create new Project” на долната фигура или да се вложи вече съществуващ чрез “or import a Google project” на фиг. (2.8):



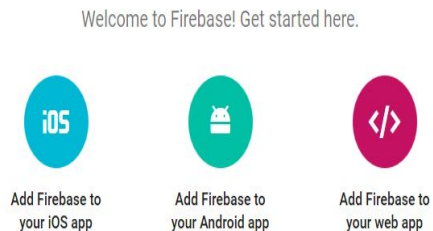
Фиг. (2.8)

- Задаваме настройките на фиг. (2.9):

The screenshot shows the 'Create a project' dialog box. It has a 'Project name' field containing 'GodOfRowing' and a 'Country/region' dropdown menu set to 'Bulgaria'. Below these fields is a disclaimer: 'By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at any time. [Learn more](#)'. At the bottom, there is a line of text: 'By proceeding and clicking the button below, you agree that you are using Firebase services in your app and agree to the applicable [terms](#)'. At the very bottom are two buttons: 'CANCEL' and 'CREATE PROJECT'.

Фиг. (2.9)

- Избираме платформа фиг.(2.10):

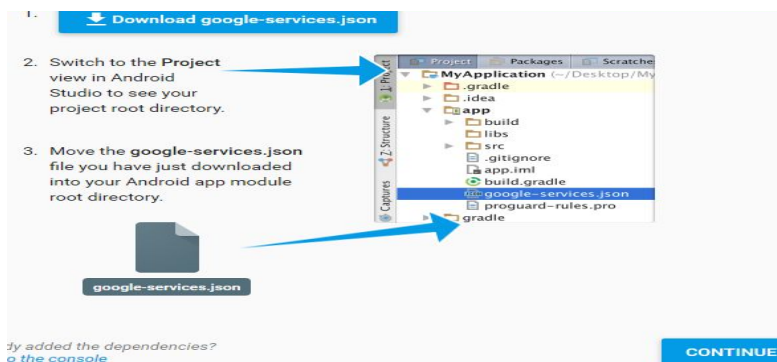


Фиг.(2.10)

- Попълваме необходимите полета и избираме “Add App” на Фиг.(2.11):

Фиг.(2.11)

- Следваме инструкциите на фиг.(2.12) - сваляме и поставяме в .app в нашия проект:  
Файла google-services.json



Фиг.(2.12)

- Добавяме зависимостите, както е посочено в [2.4]:



The Google services plugin for [Gradle](#) loads the `google-services.json` file that you just downloaded. Modify your `build.gradle` files to use the plugin.

1. Project-level `build.gradle` (<project>/`build.gradle`):

```
buildscript {
    dependencies {
        // Add this line
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

2. App-level `build.gradle` (<project>/<app-module>/`build.gradle`):

```
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'

includes Firebase Analytics by default ⓘ
```

3. Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed since last sync. Sync now

Фиг.(2.13)

- Отиваме в Authentication / sign in methods включваме Email/Password;
- Отиваме в Database / Правила за достъп(rules);  
И поставяме правилата:

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

С които указваме, че само потребители на приложението могат да използват базата от данни.

Фиг.(2.14)

### 2.1.7. Използвани сензори от Android устройството

Мобилните устройства отдавна са се превърнали от средство за изпълнение на някои прости комуникационни операции, в инструмент за много функционална употреба. Съвремените смартфони и таблети, управлявани от ОС Android, разполагат с множество сензори, които реагират на преместване, на пространствена ориентация, на различни

параметри на околната среда (магнитно поле, температура на въздуха, атмосферно налягане) и т.н. Наличието на такива сензори превръща мобилното устройство в изключително полезен, при това достъпен за потребителя инструмент. Упоредбата на сензори става както в развлекателния софтуер, така и в приложения с много по-сериозна и професионална насоченост.

Сензорът представлява електронно устройство, което е в състояние да възприема физически величини и да ги преобразува в електрически сигнали, които могат да бъдат непосредствено измервани, предавани и кодирани. Всеки сензор има свой принцип на действие, както и специфични технически параметри. Сензорите са в състояние да осигурят изходните данни (необработени “Raw data”) с необходимата прецизност и точност, да ги представят пред потребителя по подходящ начин или да ги предадат на ползващите ги софтуерни приложения.

#### **2.1.7.1. Видове сензори, поддържани от ОС Android.**

Android поддържа разнообразни сензори, които обикновено се систематизират в няколко по-обща (и донякъде условни) групи:

- Сензори за преместване (Motion sensors) – тези сензори измерват линейни и ъглови ускорения. В тази категория влизат акселерометри, сензори за гравитация, жирокопи, сензори за ротация и др.;
- Сензори, измерващи параметри на средата (Environmental sensors) – измерват разнообразни параметри на средата като например температура и влажност на въздуха, атмосферно налягане и т.н. Тук влизат различните барометри, термометри, фотометри и др.;
- Сензори за местоположение (Position sensors) – измерват физическото местоположение на устройството в географски координати. Включват сензори за ориентация и магнитометри.

По друга класификация на сензорите се разделят на

хардуерно-базирани и софтуерно-базирани.

- Хардуерно-базираните сензори са физически конструктивни компоненти, които са вградени в апаратната част на устройството. Те измерват непосредствено - конкретни физически величини. Такива сензори са сензорите за ускорение, на геомагнитното поле, за ъглови изменения, за осветеност и др.;
- Софтуерно-базираните сензори не са технически реализирани, а всъщност извличат данни от един или повече хардуерно-базирани сензори и въз основа на техните стойности изработват собствен резултат. Такива са например сензорът за гравитация, за линейно ускорение и др.

От гледна точка на съставянето на програмния код хардуерно-базираните и софтуерно-базираните сензори се интерпретират по един и същ начин.

#### **2.1.7.2. Основни технически параметри на сензорите –**

Такива са: обхват, точност, прецизност, интервал на отчитане.

#### **2.1.7.3. Използвани сензори:**

##### **2.1.7.3.1. Акселерометър**

Акселерометърът измерва ускорението по всяка от кординатните оси, с което могат да бъдат оценени силите, приложени върху мобилното устройство. Той е хардуерен сензор. Когато устройството е неподвижно, то изпитва земното ускорение от 1g (9.81 m/s<sup>2</sup>) по една от осите си или чрез векторната сума от компонентите по всички оси.

В най-общия случай акселерометърът се използва за да се определи ориентацията на мобилното устройство, но може да се използва и за измерване на линейното ускорение, както е описано в следващата точка.

Измерителната единица, използвана от сензора (в система Si) е m/s<sup>2</sup>. Следва да се вземе под внимание, че акселерометърът не измерва скорост, а интензивността на нейната промяна, което е обект на текущата разработка.

#### **2.1.7.3.2. Сензор за линейно ускорение**

Сензорът за линейно ускорение е софтуерно-базиран сензор. Той изчислява линейното ускорение по всяка от координатните оси, извличайки измерените данни от акселерометъра, от които премахва вектора на гравитацията.

### **2.1.8. Структура на проекта**

#### **2.1.8.1. Пакети на приложението**

Приложението е разделено на главни пакети, всеки от които отговаря за отделен проблем.

##### **2.1.8.1.1. Пакет `com.bzahov.elsys.godofrowing`**

Това е главният пакет на проекта. В него се съдържат всички останали пакети, както и главните Екрани (Activities).

##### **2.1.8.1.2. Пакет**

**`com.bzahov.elsys.godofrowing.AuthenticationActivities`**

В този пакет се съдържат всички необходими Activity класове необходими за Влизане и създаване на профили.

##### **2.1.8.1.3. Пакет `com.bzahov.elsys.godofrowing.Fragments`**

В този пакет се съдържат всички фрагменти, използвани в приложението.

##### **2.1.8.1.4. Пакет `com.bzahov.elsys.godofrowing.RecyclerView`**

В този пакет се съдържат класовете необходими за реализирането на RecyclerView като:

- Адаптери (Adapters)
- Държачи на изгледи (ViewHolders)

#### **2.1.8.1.5.   Пакет**

**com.bzahov.elsys.godofrowing.ResultTabContent**

В този пакет са добавени всички под-активитита на TabHost Екрана, показващ се след края на тренировка, съдържащ резултатите и или при нужда от проследяване на историята на тренировките.

#### **2.1.8.1.6.   Пакет .godofrowing.Models**

В този пакет се съдържат класовете, необходими за комуникацията с Firebase Database, които са описани в [2.4.]

### **2.2.   ОПИСАНИЕ НА ИЗИСКВАНИЯТА КЪМ ПРОГРАМНИЯ ПРОДУКТ**

#### **2.2.1.Функционални изисквания**

Създаване на мобилно приложение за „Android“ платформата, което да служи за бързо, лесно и ефективно анализиране на тренировъчния процес в реално време, както и след края на спортната тренировка с цел разбор на постигнатите резултати на спортиста от него или неговите треньори.

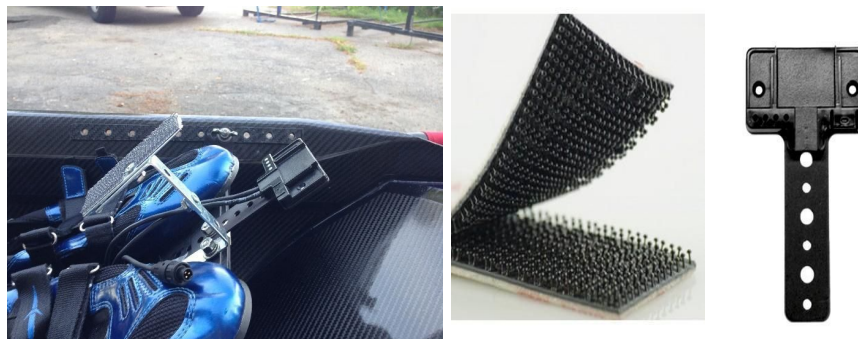
1. Отчитане на разликите в движението на устройство в пространството, посредством Акселерометъра и GPS локализиране, вградени в Android устройството;
2. Автоматично да се начертава графиката на ускорението в реално време;
3. История на тренировките, минали графики и изчисления и съпоставянето на всички данни в даден период от време, за обективен анализ на целия процес;
4. Поставяне на Android устройството в лодката.

## **2.3. Поставяне на Android Устройството**

Смартфоните са чудесен инструмент за проследяване на гребни тренировки и тяхната ефективност. Поради техните специфики е много важно да успеем да ги закрепим на сигурно място в лодката и да са максимално защитени от водата. Това може да се случи в зависимост от избора на потребителя:

### **2.3.1. Чрез самоделна установка**

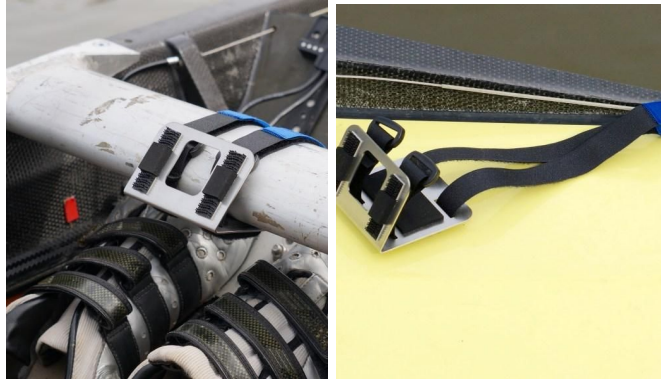
Устройството може да бъде поставено в самата лодка на “крачното”, където гребецът поставя краката си, за да има опора. Прикрепяме го за основата на крачното, посредством метална пластинка и винтове. Тази пластина има плосък край, на който залепваме едната част от двустенна лепка. А на водонепромокаемия калъф залепваме другата част от лепенката.



Фигури (2.15-2.18) Крачно и приспособление за закрепяне

### **2.3.2. Чрез Rowing Boat Smartphone Mount**

Чрез тази поставка, устройството може да бъде поставено почти навсякъде в лодката, може да бъде прикрепен към крилата, ако те са от вида показан по долу на Фигура (2.19) или на палубата Фигура(2.20). Тази поставка е направена от твърда, неръждаема стомана с внимателно изградена прикрепяща част, която гарантира, че няма да бъде надраскан корпусът на лодката. Чрез нея устройство е поставено на неподвижна и сигурна основа.



Фигури(от ляво 2.19 и от дясно 2.20)

## 2.4. Описание на спомагателните класове за Firebase Database

Чрез тези класове, намиращи се в пакета `godofrowing.Models` се обработва информацията подаваща се, както и извличаща се от облака. Това е алтернатива на работата с множество от данни, пооделно (всяка с отделна заявка), което би затруднило програмистта, както и би направило кода неразбираем и много по-обемен.

Чрез “`@IgnoreExtraProperties`” се “информира” FireBase api-то, че трябва игнорира полетата на класа, които не се използват за достъпване на отделните полета (Get-ери).

### 2.4.1. Класът за потребителската информация (User)

Този клас се подава на базата от данни и с негова помощ, тя извлича/записва информацията за потребителя.

За потребителя се запазва следната информация:

- потребителско име;
- идентификационен номер(автоматично генериран от api-то);
- имейла на потребителя;
- идентификационен номер на клуб;
- списък (List) с всичките му спортни дейности:

```
@IgnoreExtraProperties
public class User { // Data Class for Send information
```

```
private String username;
private String email;
private String age;
private int userClubId;
```

```

private int uid;
private List<ResourcesFromActivity> resourcesFromActivityList; //по-долу

public User() {
    // Default constructor required for calls to
    DataSnapshot.getValue(User.class)
}
public String getUsername() {
    return username;
}
// (...) other's getters and constructors
}

```

### 2.4.2. Класът съдържащ информацията от тренировката

Този клас се подава на базата от данни и с негова помощ, тя извлича/записва данните от изминалата тренировка.

```

@IgnoreExtraProperties
public class ResourcesFromActivity {
    private String currentTime;
    private long elapsedTime;
    private String elapsedTimeStr; //00:09:12
    private float averageSpeed;
    private float maxSpeed;
    private float averageStrokeRate;
    private long totalMeters;
    private List<MyLocation> myLocationsList = new ArrayList<>();
    private List<Float> allStrokes = new ArrayList<>();

    public ResourcesFromActivity(){
        // Default constructor required for calls to
        DataSnapshot.getValue(ResourcesFromActivity.class)
    }
}

```

### 2.4.3. Реализиране На MyLocation class

- Необходим е, защото оригиналният клас Location няма конструктор по подразбиране (Default), и класът DataShapshot не може да го използва.

```

@IgnoreExtraProperties
public class MyLocation {

    private long time;
    private double lat;
    private double lng;
    private float speed;
    public MyLocation() {}
}

```



### 3. ТРЕТА ГЛАВА.

#### ОПИСАНИЕ НА РАБОТАТА ПО АНДРОИД ПРИЛОЖЕНИЕТО.

Приложението е разделено на пакети, като класовете във всеки пакет отговарят за определени проблеми.

##### 3.1. **Пакет** `com.bzahov.elsys.godofrowing.AuthenticationActivitie`

В този пакет се съдържат екраните, които са необходими за влизането и регистрирането на потребители.

##### 3.1.1. Реализация формата за вписване (Login)

Реализацията на Login опцията започва с дефинирането на `Public class LoginActivity`, който наследява класа `AppCompatActivity`.

```
public class LoginActivity extends AppCompatActivity {
```

Когато се създава екрана в `onCreate` се задава кое оформление (layout) да бъде използвано. В случая се използва **“activity\_login”**

```
@Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_login);
```

Ако нямате акаунт може да се регистрирате като натиснете бутона Sign Up и той ще ви отведе до `SigUpActivity`, използвайки `Intent`, по следния начин:

```
btnLinkToSignUp.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Intent intent = new Intent(LoginActivity.this, SignInActivity.class);  
        startActivity(intent);  
    }  
});
```

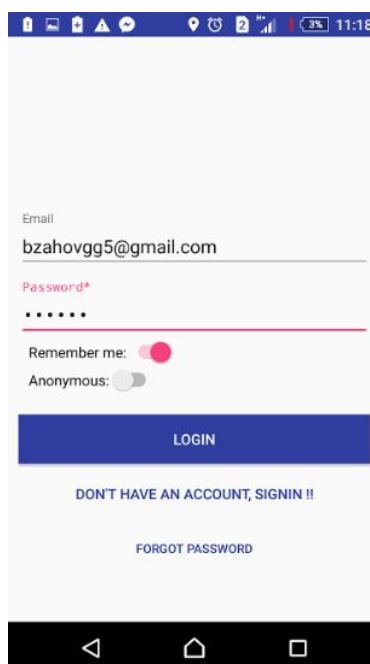
Ако натиснете бутона Login, то ще се вземе съответно името и паролата, която сте подали и в текстовите полета и ако са валидни ще се изпрати заявка към Firebase Authentication, посредством `Api` –то на Firebase, извиквайки функцията `signInWithEmailAndPassword` с вече

проверените, от към формат, низове емайл и парола. Ако има грешка от сървърната страна, като например несъществуващ потребител и/или грешна парола, потребителя ще трябва да се идентифицира отново. При успешна заявка, потребителят ще се върне към предходния екран, след извикване на `final()`, който прекратява сегашния екран.

```
mAuth = FirebaseAuth.getInstance();
mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {

            progressBar.setVisibility(View.GONE);
            if (!task.isSuccessful()) {
                // има грешка
                Toast.makeText(LoginActivity.this, getString(R.string.auth_failed),
                    Toast.LENGTH_LONG).show();
                Log.e(TAG, "signInWithEmail", task.getException());
            } else {
                Intent returnIntent = new Intent();
                returnIntent.putExtra("returnFromLogin", false);
                setResult(Activity.RESULT_OK, returnIntent);
                finish();
            }
        }
    });
```

Има възможност да продължите без да се регистрирате или влизате във вече съществуващ профил, като изберете ключа `Anonymous(Switch)`:



фиг.(3.1)

### 3.1.2. Реализация на форма за регистриране (SignIn)

Реализацията на SignIn опцията започва с дефинирането на Public class SignInActivity, който наследява класа AppCompatActivity.

Фиг.(3.2.)

```
public class SignInActivity extends AppCompatActivity {
```

Ако имате акаунт може да се влезете в него, като натиснете бутона “Already Registered” и той ще ви отведе до SigUpActivity, използвайки Intent, по следния начин:

```
btnLinkToLogIn.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Intent intent = new Intent(SignInActivity.this, LogInActivity.class);  
        startActivity(intent);  
    }  
});
```

Ако натиснете бутона SignUp, то ще се вземе съответно името и паролата и другите не задължителни полета, които сте подали и в текстовите полета и ако са валидни ще се изпрати заявка към Firebase Authentication, посредством Api –то на Firebase, извиквайки функцията `createUserWithEmailAndPassword` с вече проверените, от към формат, низове емайл и парола. С останалите незадължителни полета и email-а Ще се създаде нов клон в базата от данни, чрез `mDatabase.child("users").child(userId).setValue(newUser);`

- където `mDatabase = FirebaseDatabase.getInstance().getReference()` е препратка към главното разклонение (root)
- `.child("users")` указва препратка към разклонението с потребителите,
- `.child(userId)` указва препратка към разклонението на конкретния потребител, където `userId` е получено при успешно създадения акаунт -`userFromRegistration.getUid()`;

След това 'чрез тази препратка се посочва къде да запише информацията съдържаща се в обекта `newUser`.

Ако има грешка от сървърната страна потребителят ще трябва да се регистрира отново. При успешна заявка, той ще се върне към предходния екран, след извикване на `final()`, който прекратява сегашния екран.

```
auth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(SignInActivity.this, new OnCompleteListener<AuthResult>()
    {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            Log.d(TAG, "createUserWithEmail:onComplete:" + task.isSuccessful());
            progressBar.setVisibility(View.GONE);

            if (!task.isSuccessful()) {
                Log.d(TAG, "Authentication failed." + task.getException());
            } else {
                createNewUser(task.getResult().getUser());
                finish();
            }
        }
    });
```

### 3.1.3. Пакет `com.bzahov.elsys.godofrowing`;

Това е главният пакет. В него са поместени всички останали под пакети, както и двата основни екрана.

### 3.1.4. Реализация на основния екран MainActivity

Това е главният екран, на който се изобразяват всички параметри и графики на текущата спортна дейност. Получава главна част от информацията Класът наследява `FragmentActivity`. Също така имплементира няколко канала за получаване на данни от различните Фрагменти.

```
public class MainActivity extends FragmentActivity implements
MainMapFragment.MapFrgCommunicationChannel,
MainGraphFragment.GraphFrgCommunicationChannel, LinAccelFrgCommunicationChannel{
```

Това е главният екран, от който стартира и самото приложение (зададен в „Manifest“, предоставящ съществена информация за приложението на системата Android):

```
<activity
    android:name="com.bzahov.elsys.godofrowing.MainActivity"
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Оформлението на този екран представлява `LinearLayout`, който съдържа `TableLayout`, `HorizontalScrollView` и `FrameLayout`. `LinearLayout`, указва чрез `weightSum` атрибута на колко парчета се равни 100% от екрана. В случая той е зададен на 3. Всеки от горе изброените оформления приема размера на зададения му `layout_weight`. Например Табличното оформление е с `layout_weight` 1.6, което означава, то ще заеме малко повече от половината.

Нека започнем с оформлението “TableLayout“, което представлява таблица с редове, изградени от под оформления, в които се намират текстовите полета за изобразяване на параметрите, както и техните мерни единици.

Под него се намира хоризонтално-плъзгащ се изглед. Той ни позволява да имаме повече изгледи, от колкото екрана може да побере. Задължително е да има само едно оформление, като например “Relative Layout“, в което са добавени различни изгледи, до които се достига чрез плъзгане. При натискане на някое от тях се зарежда фрагмент в долния изглед, чрез OnClick Listener, засичащ натискването избрания фрагмент и го показва (създава, ако не съществува още), скривайки останалите.

```
public void ScrollItem(View view) {  
    switch (view.getId()) {  
        case R.id.param_2:  
            choosedDetailOption = 2;  
            displayFragment(MAP_FRAGMENT);  
            break;  
        (...)  
    }  
}
```

Да преминем към третата част, изграждаща главния екран – „FrameLayout“, той представлява рамково оформление, позволяващо координирането на поведението и състоянието на своите “деца” (children) при чести действия с тях. В конкретния случай оформлението съдържа фрагменти. Първо трябва да изясним, какво е фрагмент:

Фрагментът е самостоятелен компонент със собствен потребителски интерфейс и жизнения цикъл. Той може да се използва многократно в различни части на потребителския интерфейс на приложението, в зависимост от желанния случай за конкретно устройство или екран. В някои отношения може да се мисли за един фрагмент, като за мини-екран, макар че той не може да работи самостоятелно, а само в рамките на вече съществуващ екран (Activity).

Както вече беше написано по-горе, след избор на услуга от

ползвателя (от плъзгащото се меню), се създава, показва или скрива дадения фрагмент, с помощта на транзакции (fragmentTransaction)

Първо е необходимо да се обясни какво е FragmentManager и транзакция

- FragmentManager управлява Фрагменти в Android, по-точно - обработва транзакции между фрагменти.
- Транзакцията е средство, с което се добавя, заменя или премахва фрагмент.

Операциите са извършени с тези методи:

```
FragmentTransaction frgTransact=getSupportFragmentManager().beginTransaction();
```

```
fragmentTransaction.add(R.id.details, fragment, fragment.getTag()); - добавя нов  
фрагмент в контейнера R.id.details  
fragmentTransaction.remove(fragment); - премахва посочения фрагмент  
fragmentTransaction.show(fragment); - показва посочения фрагмент  
fragmentTransaction.hide(fragment); - скрива посочения фрагмент  
fragmentTransaction.commitNow(); -прилага промените
```

Фрагментите които са на разположение:

### 3.2.1.1 Абстрактният фрагмент за графика BaseChartFragment

Този фрагмент се наследява в приложението, когато трябва да се начертава графика. Наследява класа Fragment пренаписва метода onCreateView. Например наследява се от фрагмента за изобразяване на линейното ускорение.

```
public abstract class BaseChartFragment extends Fragment {  
(...)  
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle  
savedInstanceState) {  
  
View v = inflater.inflate(R.layout.fragment_graph, parent, false);  
setGraph(v);  
}
```

```

    setSensor();
    (...)
}
return v;
}

```

В него са имплементирани необходимите методи за създаване на графика:

- setGraph;
- addEntry;
- createSet;
- addDataSet;
- removeDataSet.

Наследниците на този клас е необходимо само да извикват методите му, за да създадат и попълнят графика.

### 3.1.5. Фрагмент за изчисляване на линейното ускорение

Този клас наследява BaseChartFragment, за да може да рисува графики.

Действията са подобни за останалите фрагменти от този тип и затова ще разгледаме само този клас.

```

public class MainLinAccGraphFragment extends BaseChartFragment implements
SensorEventListener {

```

Имплементира SensorEventListener, за да може да чете от вградените сензори в телефона. Ако устройството разполага със сензора за линейно ускорение (Sensor.TYPE\_LINEAR\_ACCELERATION), взима директно данните от него и използвайки методите на BaseChartFragment ги начертава. Ако обаче устройството не разполага с този сензор, чете данните от акселерометъра (Sensor.ACCELEROMETER) и ги обработва, за да се получи линейното ускорение.



### 3.1.5.1. Регистриране на сензорите

Първо трябва да направим инстанция на сензор-мениджъра, както е показано на 3-тия ред.

След това трябва да изберем типа сензор, и да регистрираме Listener, който да ни уведомява, кога има нови данни.

```
@Override
protected void setSensor() {
    mSensorManager=(SensorManager) getContext().getSystemService
(Context.SENSOR_SERVICE);
    //Linear Acceleration sensor

    mLinAcceleration=mSensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);

    mSensorManager.registerListener(this,mLinAcceleration,SensorManager.SENSOR_DELAY_NORMAL
);
    //Accelerometer sensor
    mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    mSensorManager.registerListener(this, mAccelerometer ,
SensorManager.SENSOR_DELAY_NORMAL);
}
```

### 3.1.5.2. Изчисляване на линейното ускорение

В случай, че имаме сензор за линейно ускорение на устройството си, директно взимаме данните му, от метода onSensorChanged, който се извиква след като сме се регистрирали за нови данни.

Първо проверяваме дали onSensorChanged е извикано за нашия сензор и ако да взимаме данните от трите оси на сензора и прилагаме питагоровата теорема за да намерим цялостното ускорение. След това добавяме новото ускорение към графиката чрез addEntry.

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    Sensor accSensor = sensorEvent.sensor;
    if (accSensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
        Log.d("LinAccel", "has ");
        x_linear_acceleration = sensorEvent.values[0];
        x_linear_acceleration = sensorEvent.values[1];
        z_linear_acceleration = sensorEvent.values[2];
    }
}
```

```

        if (x_linear_acceleration == 0 && y_linear_acceleration == 0 &&
z_linear_acceleration == 0){
            return; // does not have Sensor.TYPE_LINEAR_ACCELERATION
        }

        currentAcceleration = (((float) Math.pow(x_linear_acceleration, 2)) +
(float)Math.pow(y_linear_acceleration, 2)+ (float)Math.pow(z_linear_acceleration, 2));

        addEntry(lineGraphChart, currentAcceleration, 2);
    }

```

Ако устройството ни не разполага със сензор за линейно ускорение, го изчисляваме чрез данните от акселерометъра по следния начин:

Акселерометър може да се измерва статичното гравитационно поле на земята (като сензор за наклон) или може да се измери мярката линейното ускорение (като ускоряване на лодката), но той не може да измери и двете по едно и също време. Когато говорим за линейно ускорение във връзка със сензор за ускорение (акселерометъра), ние може да намерим линейно ускорение, което е равно на Измереното Ускорение, от което е изваден векторът на Гравитацията (така че ние можем да определим действителното ускорение на устройството, без значение как устройството е ориентирано / наклонено). Трудната част е да се определи каква част от сигнала е гравитацията.

Силата на гравитацията може да бъде изолирана от Low-pass филтър.

Периодът се изчислява с помощта на метода на осредняване. Това се прави, защото на периода между две непосредствени опреснявания, на данните, е непоследователен (sampleTimeNew - sampleTimeOld). Методът на осредняване дава по-добра оценка на периода.

```

if (accSensor.getType() == Sensor.TYPE_ACCELEROMETER) {
    System.arraycopy(sensorEvent.values, 0, this.accelerationOnAxis, 0,
sensorEvent.values.length);

    x_accelerometer = sensorEvent.values[0];
    y_accelerometer = sensorEvent.values[1];
    z_accelerometer = sensorEvent.values[2];
}

```

```

timestamp = System.nanoTime();

float deltaTime = timestamp - timestampOld;

dt = 1 / (count / ((deltaTime) / 1000000000.0f));

count++;

alpha = timeConstant / (timeConstant + dt);

x_gravity = alpha * x_gravity + (1 - alpha) * x_accelerometer;
y_gravity = alpha * y_gravity + (1 - alpha) * y_accelerometer;
z_gravity = alpha * z_gravity + (1 - alpha) * z_accelerometer;

x_linear_acceleration = x_accelerometer - x_gravity;
y_linear_acceleration = y_accelerometer - y_gravity;
z_linear_acceleration = z_accelerometer - z_gravity;

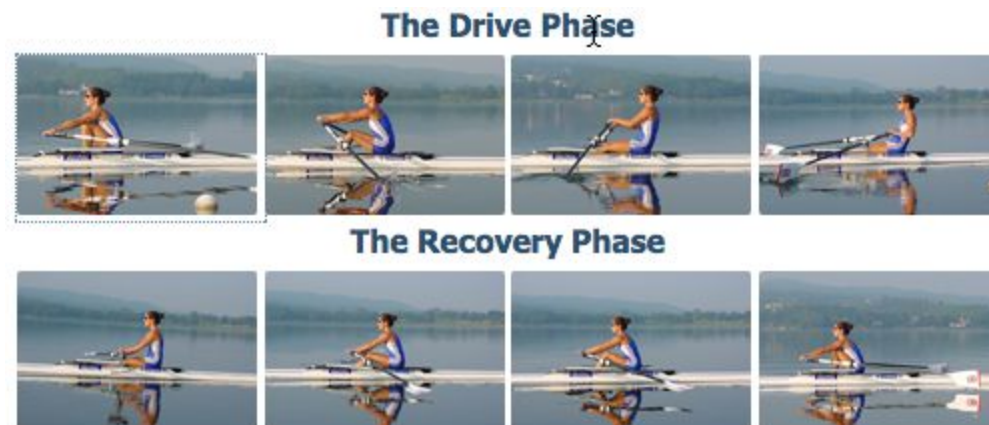
}

currentAcceleration = (float) (Math.pow(x_linear_acceleration, 2) +
Math.pow(y_linear_acceleration, 2) + Math.pow(z_linear_acceleration, 2));
allLinAccelData.add(currentAcceleration);

```

### 3.1.5.3. Засичане на нов загребок

Първо трябва да изясним, какво е загребок в Академичното Гребане - това е един пълен цикъл на движението в спортното гребане, който е показан на фигурата:

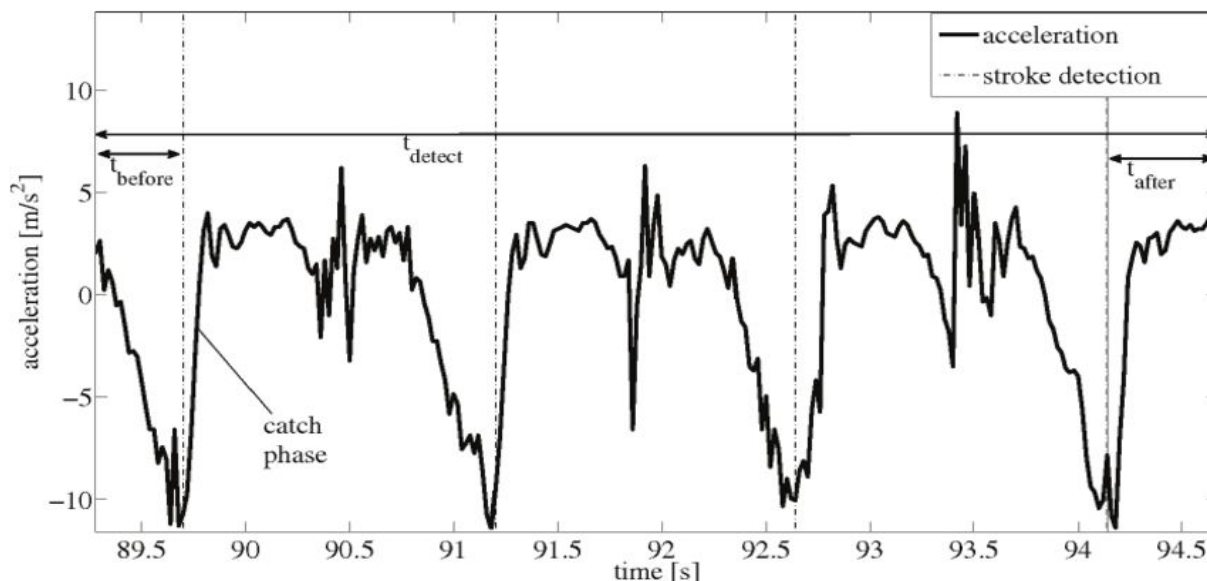


Фигура (3.3)

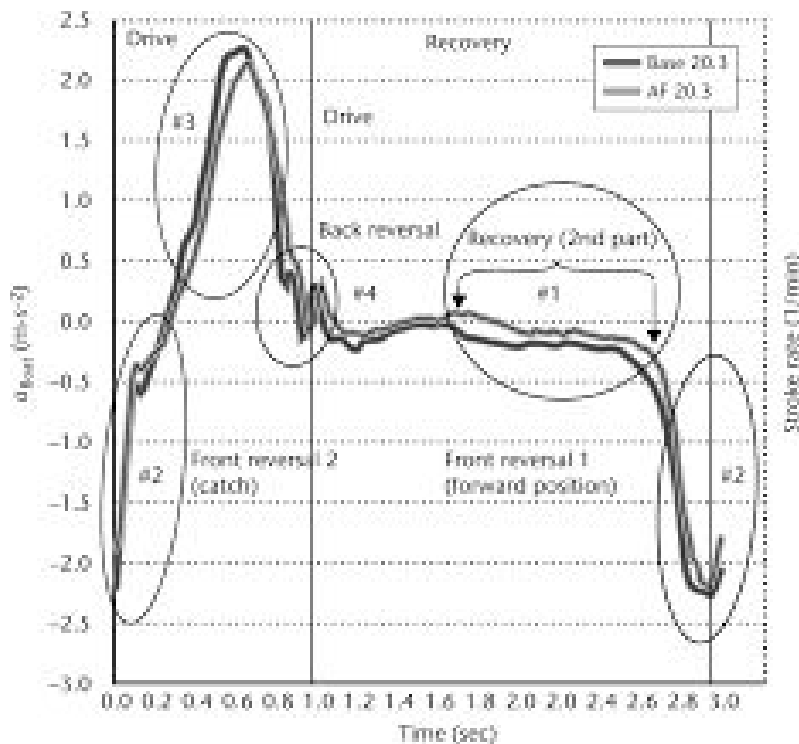
Нов загребок е (на фигура (3.4) с пунктирна линия се изобразява всеки нов загребок)

Началото на нов загребок е моментът в който гребеца се е свил в предната част от лодката (първия период, първата снимка на Фигура 3.3, а на долната фигура (Фигура 3.4) е изобразено ускорението в  $m/s^2$  към изминатото време. Тук новия загребок е отбелязан с вертикална пунктирна линия).

За да се измери, кога има нов загребок се използва следната логика: Ако ускорението е отрицателно, се проверява дали вече не се ускорявало достатъчно продължително време.



Фигура (3.4)



Фигура 3.5

#### 3.1.5.4. Комуникацията между Фрагмент и Екран

Най-лесният начин за комуникация между фрагмент и екран е да се използват интерфейси.

Първо трябва да направим интерфейс:

```
public interface LinAccelFrgCommunicationChannel {
    void notifyForNewStroke();

    void sendNewLimAccel(float l);
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof LinAccelFrgCommunicationChannel) {
        mCommChListner = (MainLinAccGraphFragment.LinAccelFrgCommunicationChannel) context;
    } else {
        throw new ClassCastException();
    }
}
```

Извикваме този метод в фрагмента и в Екрана го обработваме

```
private void setnotifyForNewStroke() {  
    //communication with activity  
    mCommChListner.notifyForNewStroke();  
    Toast.makeText(getApplicationContext(), "New Stroke", Toast.LENGTH_SHORT).show();  
}
```

В екрана(Activity) се извиква този, когато сме изпълнили setnotifyForNewStroke()

```
@Override  
public void notifyForNewStroke() {  
    strokeCounterRate();  
}
```

### 3.1.6. Фрагментът с Google карта

```
public class MainMapFragment extends Fragment implements OnMapReadyCallback,  
LocationListener, GoogleApiClient.ConnectionCallbacks, OnConnectionFailedListener {
```

Този фрагмент се използва за да се покаже пътят, който състезателят е изминал и когато кликне на маркер от картата да вижда какви са били показателите му в дадения момент. По време на работата на проложението се изисква от потребителя да включи GPS локализирането на своето устройство, ако не го е направил предварително. Когато се включва програмата за първи път и/или потребителят не е дал разрешение, за гео-локализирането на устройството, такова се изисква, чрез диалог. Локализирането става чрез GPS provider.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {  
    checkLocationPermission();  
} else {  
    if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {  
        showSettingsAlert();  
    }  
}  
  
private boolean checkLocationPermission() {  
    String[] permStrArray = new String[]{  
        Manifest.permission.ACCESS_FINE_LOCATION,  
    };  
    Log.d(TAG, "CheckLocationPermissions");  
    if (ContextCompat.checkSelfPermission(getApplicationContext(),
```

```

Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    if (ActivityCompat.shouldShowRequestPermissionRationale(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION)) {
        ActivityCompat.requestPermissions(getActivity(), permStrArray,
MY_PERMISSION_REQUEST_LOCATION);
    } else {
        ActivityCompat.requestPermissions(getActivity(), permStrArray,
MY_PERMISSION_REQUEST_LOCATION);
    }
    return false;
} else {
    Log.d(TAG, "HavePermission");
    if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        showSettingsAlert();
    }
}
return true;
}

@Override
public void onLocationChanged(Location location) {

```

### 3.1.7. Изпращане на дейността от тренировката

След като информацията е събрана и обработена и тренировката е приключила, тя трябва да се качи в базата от данни. Това става чрез функцията `SendDataToDatabase()`.

Първо - Трябва да се създаде нов обект виж **ResourcesFromActivities(2.3.2)**

Второ - Необходимо е да се вземе препратка към разклонението където се запазват данните в базата.

Трето - Трябва да се качи информацията, чрез `setValue()`:

```

private void SendDataToDatabase() {
    DatabaseReference myRef = database.getReference();

    String currentDateTimeString = DateFormat.getDateTimeInstance().format(new
Date()).replaceAll("[, . ]", "");
    ;
    DateFormat dateFormat = new SimpleDateFormat("yyyy:MM:dd:HH:mm:ss");
    DateFormat dateFormatWrap = new SimpleDateFormat("yyyyMMddHHmmss");
    String postTime = dateFormatWrap.format(new Date());

```

```

String postTimePresentation = dateFormatWrap.format(new Date());

ResourcesFromActivity rfa = new
ResourcesFromActivity(totalMeters, averageStrokeRate, maxSpeed, averageSpeed, elapsedTimeSt
r, postTime);
DatabaseReference dataRef = database.getReference().child("users/" + mUser.getUid() +
"/activities/" + currentDateTimeString);
dataRef.setValue(rfa);
}

```

## 3.2. Екрана разпределител ResultActivity

Този екран представлява навигатор между различните екрани след завършване на тренировката, както и този с историята. Наследява TabActivity, при който вид екран има възможност да се сложат табове, а при натискането на всеки един под тях се зарежда специфичен екран.(activity)

```

public class ResultActivity extends TabActivity {
    TabHost host;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // setContentView(R.layout.activity_result);

        host = getTabHost();

        host.addTab(host
            .newTabSpec("Tab for analysis")
            .setIndicator("", getResources().getDrawable(R.drawable.icon_analysis, null))
            .setContent(new Intent(this, ResultContentAnalysisActivity.class)));

        host.addTab(host
            .newTabSpec("Tab for time Split")
            .setIndicator("", getResources().getDrawable(R.drawable.icon_timer, null))
            .setContent(new Intent(this, ResultContentSplitterActivity.class)));

        host.addTab(host
            .newTabSpec("Tab for settings")
            .setIndicator("", getResources().getDrawable(R.drawable.icon_settings, null))
            .setContent(new Intent(this, ResultContentHistoryActivity.class)));
        host.setCurrentTab(0);
    }
}

```



```
host.getTabWidget().setBackgroundColor(getResources().getColor(R.color.wallet_holo_blue_light));
```

### 3.3. Екранът за резултатите от тренировката

В този екран се извлича информацията от базата за последната спортна тренировка и тя се представя в четим, за потребителя, формат.

```
public class ResultContentAnalysisActivity extends Activity implements
    OnMapReadyCallback, ValueEventListener {

    (...)
    myQuery.addValueEventListener(this); // Query
    (...)
    private void setParameters(int viewID, int imageID, @Nullable String name, String
value) {
        RelativeLayout viewById = ((RelativeLayout) analysisContainer.findViewById(viewID));
        if (imageID != 0) {
            ImageView imageView = ((ImageView)
viewById.findViewById(R.id.res_layout_parameter_image));
            imageView.setImageResource(imageID);
        }
        if (name != null) {
            TextView nameView = ((TextView)
viewById.findViewById(R.id.res_layout_parameter_name));
            nameView.setText(name);
        } if (value != null) {
            TextView valueView = ((TextView)
viewById.findViewById(R.id.res_layout_parameter_value));
            valueView.setText(value);
        }
    }
}
```

### 3.4. Екранът за историята на всички тренировки

В този екран се извлича информацията от базата за всички изминали спортни тренировки. Изобразява се чрез `FirestoreRecyclerView`, за което е необходимо да се имплементира Адаптер и клас обработващ `ViewHolder`.

Първоначално се изобразяват, само датата на която се е провела тренировката, както и изминатите метри (заглавната част с основната информация), за да може потребителят да се ориентира и намери търсената тренировка. След натискане на заглавната част се показва под-изгледа (`child`). В него информацията е форматирана и представена както в **ResultContentAnalysisActivity(3.3)** се представя в четим, за потребителя, формат.

```
private void setupRecyclerview() {
    RecyclerView recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

    mMyAdapter = new FirestoreRecyclerViewAdapter<ResourcesFromActivity,
    FirestoreResViewHolder>(ResourcesFromActivity.class,
    R.layout.category_history_list_item, FirestoreResViewHolder.class, mListItemRef) {

        @Override
        public void populateViewHolder(FirestoreResViewHolder resourcesViewHolder,
        ResourcesFromActivity resourcesFromActivity, int position) {

            resourcesViewHolder.setKey(getRef(position).getKey());
            resourcesViewHolder.bindSportActivity(resourcesFromActivity);

        }
    };
    recyclerView.setAdapter(mMyAdapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
}
```

## **4. ЧЕТВЪРТА ГЛАВА.**

### **ИЗИСКВАНИЯ. РЪКОВОДСТВО НА ПОТРЕБИТЕЛЯ**

За да улеснение и максимално ефективна работата на потребителя с приложението, ще разгледаме поотделно всеки един екран и възможностите, които той предлага, както и ще изясним изискванията за използване на приложението.

#### **4.1. Изисквания**

За стартиране на приложението е необходимо наличието на Андроид устройство (Android device) с минимална версия 5.0 Marshmallow (Api level 23). За нормалната работа с приложението е необходимо наличието на хардуерните GPS и Accelerometer сензор.

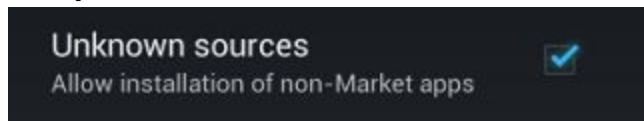
Също така е желателно потребителят да разреши, по време на изпълнението на програмата (Runtime Permissions) използването на следене на местоположението. Необходимо е разрешение за достъп до интернет, но е достатъчно то да е декларирано в Manifest.xml

#### **4.2. Инсталация**

##### **4.2.1. През електронна поща**

Можете да инсталирате приложението на Андорид устройството си по електронната поща:

1. Напишете писмо до адреса на електронната поща, който използвате на мобилното си устройство и може да прикачите към него app-release.apk файла.
2. За да можете да инсталирате приложението обаче трябва да разрешите инсталацията от неизвестни източници от: Settings>Security>Allow installation of non-Market apps.



Фиг (4.1)

3. Отворете писмото от мобилното си устройство и инсталирайте

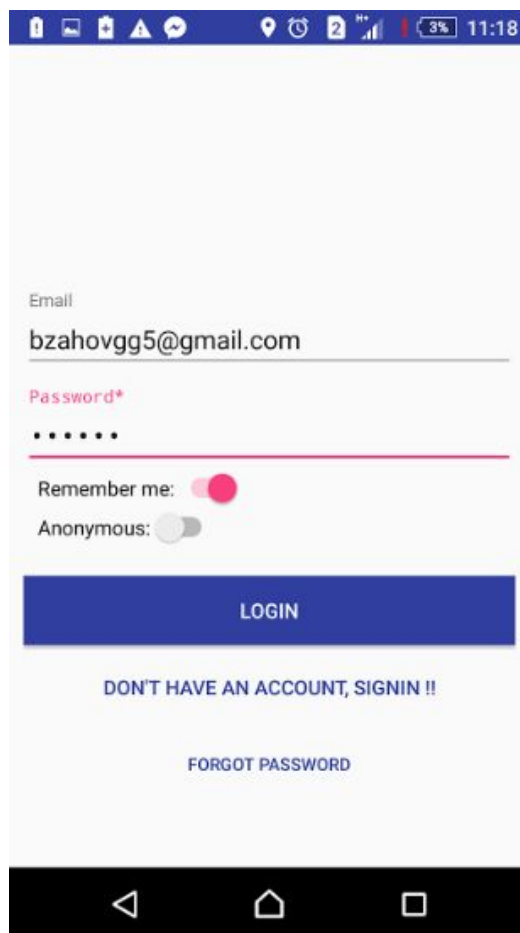
файла.

#### 4.2.2. През Google Play

Евентуално ще бъде добавен, когато приложението влезе в употреба.

### 4.3. Инструкции за използване

#### 4.3.1. Влизане в профил



Фиг(4.2)

##### 4.3.1.1. Чрез собствен Акаунт

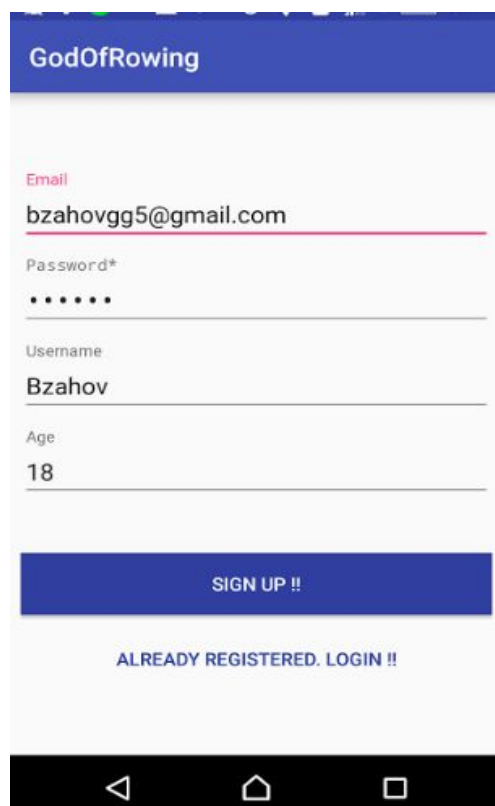
Ако потребителят има вече съществуващ профил, той може да влезе в него, като попълни полетата с email адреса и паролата си. След това той трябва да натисне бутона “LOGIN”. Ако

няма профил той може да натисне “Don’t hava an account” и ще бъде препратен до екрана с регистрирането, което е обяснено в **4.3.2** Регистриране.

#### **4.3.1.2. Чрез анонимен Акаунт**

Потребителят може да натисне ключа за Анонимно влизане. Тогава той може да ползва приложението, но няма да разполага с история на тренировките.

#### **4.3.2.Регистриране**



Фиг(4.3)

Регистрирането става през екрана за регистрация в базата от данни.

Потребителя трябва да въведе следните полета задължително:

- Email
- Парола

,а тези пожелание:

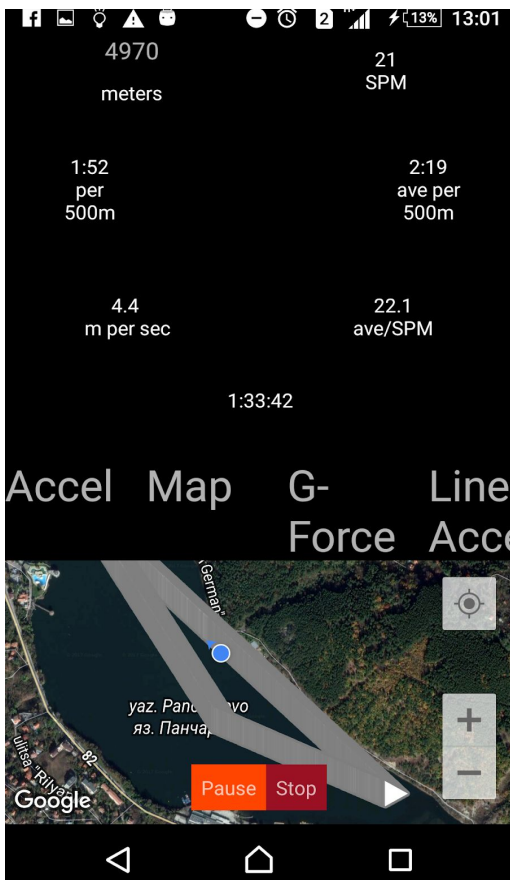
- Име на профила
- Възраст

След това трябва да се натисне бутона “SignUp”. Ако са въведени верни данни, регистрацията е направена

#### 4.4. Запознаване с основните параметри, които се отчитат в основния екран

Приложението отчита следните спортни параметри:

- изминатите метри в (meters)
- Загребоци в минута (SPM - Strokes per Minute)
- Средноаритметично на загребозите в минута(ave SPM)
- Скорост във време за изминаване на 500 метра (per 500m)
- Средна скорост във време за изминаване на 500 метра (ave per 500m)



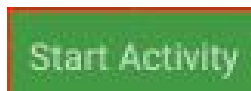
Те са показани в най горната част от основния екран на фигурата от ляво (4.4). Под параметрите се намира плъзгащо меню в което се избира, какъв допълнителен екран да се покаже.

На допълнителния екран с картата се изобразява карта, на която са нанесени маркери показващи изминалия път. При натискане на някой от маркерите, над него се появява поле, в което се показват параметрите, отчетени в този момент.

На другите допълнителни екрани се показва графика, с отчетените и изчислени данни, според избрания в плъзгащото меню елемент.

#### 4.5. Започване на записването на тренировката

За да започне записването на тренировката е необходимо потребителят да натисне зеленият бутон “Start Activity”



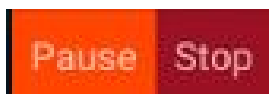
Фиг.(4.5)

##### 4.5.1. Бутона за започване на тренировката Start Activity

След натискането на този бутон, записването на тренировката започва автоматично, като в най долното поле се появява Карта, на която се начертава изминатия път. При натискане на някой от маркерите се извежда информацията за конкретното местоположение, като например с каква скорост се е движил. Също така ще се появят два бутона, където е бил “Start Activity” с два цвята - оранжев за временно прекратяване (4.5.2) и червен бутон за прекратяване на тренировката Stop Activity (4.5.3)

##### 4.5.2. Бутона за временно спиране Pause

В случай, че потребителят иска да прекъсне измерването за определен период от време, той трябва да натисне оранжевия бутон “Pause”, чрез който записването на данните ще се прекрати временно, но не и самото отчитане. Веднага след това се появява зеленият бутон “Resume”



Фигура(4.6)

##### 4.5.3. Бутона за продължаване Resume

След натискане на този бутон, записването на тренировката ще продължи нормално, без паузата да се е отразила на резултатите



Фигура(4.7)

#### 4.5.4. Бутон за прекратяване Stop

При натискането на червения бутон “**Stop**”, записването се прекратява и се отваря нов екран, където са представени резултатите от тренировката. Също така всичко записано се качва в историята на тренировките, в настоящия профил на потребителя. Резултатите от тренировката могат да бъдат проследени чрез **Резултати от текущата тренировка(4.3.5)**

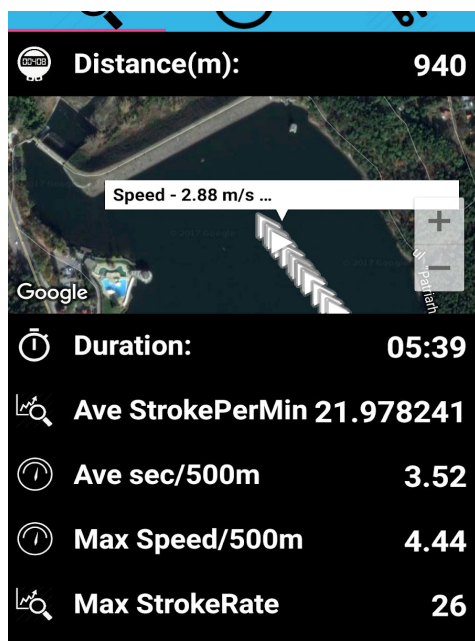


Фигура (4.8)

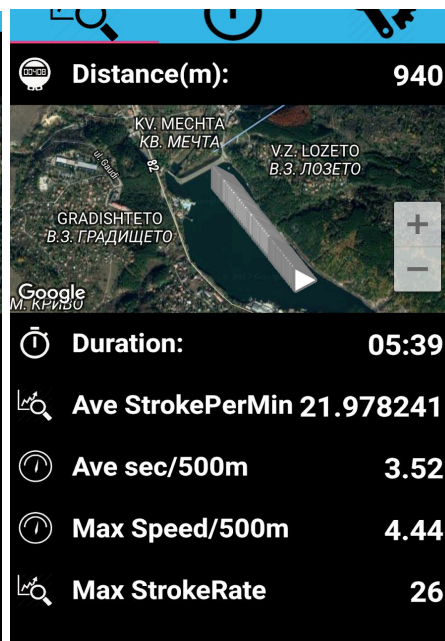
#### 4.5.5. Резултати от текущата тренировка

На този екран се изобразяват резултатите и измерените параметри от изминалата тренировка. На картата може да се избере маркер и при натискането му може да се разбере какви са били параметрите в конкретния момент.





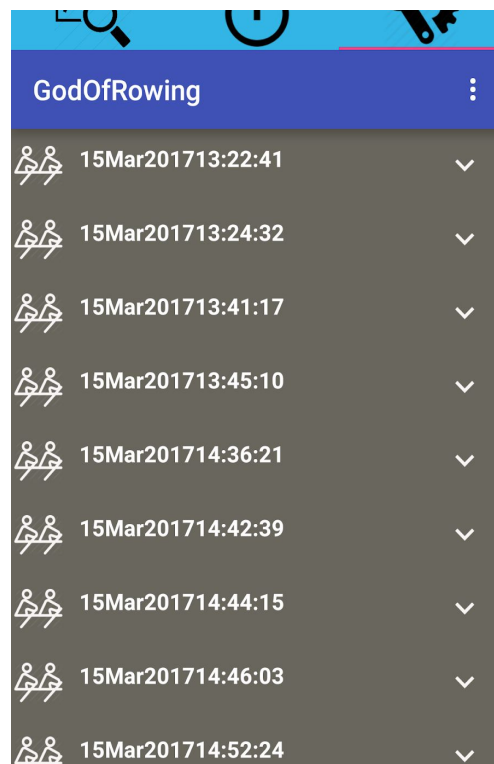
Фиг (4.8)



Фиг (4.9)

#### 4.5.6. Преглед на историята на тренировките

На този екран се изобразява основната информация кога се е провела дадена тренировка в RecyclerView. При натискане на дадената тренировка се отваря екран като **4.5.5 Резултати от текущата тренировка**, където се показват подробности



Фиг(4.10)

## **5. ПЕТА ГЛАВА**

### **Заключение**

#### **5.1. Постигания в дипломната работа**

Разработеното приложение покрива минималните изисквания и поставените задачи.

#### **5.2. Виждане за усъвършенстване на разработката**

За бъдещо усъвършенстване на разработката е необходимо да се подобри потребителския интерфейс, като се направи по-удобен.

Да се добавят още функционалности, с които да се подобри приложението и способността му да помага на треньори и спортисти в по време на тренировка( реално време) и/или след самата тренировка.

Да се добави функционалност, чрез която да се следи в реално време тренировката на друго устройство. Така например треньорите ще следят и контролират своите състезатели, на момента.

Свърване на приложението със системата на Concept2 API, която е платформа за тренировки извън водата, на специални аргометри, с програмируем компютър, чрез тяхното API. и други.

## Използвана Литература:

- [1]<http://archive.usrowing.org/about/rowing101>
- [2][http://rowingcanada.org/sites/default/files/rca\\_technique\\_final.pdf](http://rowingcanada.org/sites/default/files/rca_technique_final.pdf)
- [3]<https://developer.android.com/guide/index.html>
- [4]<https://itstep.bg/blog-bg/zashho-java-e-edin-ot-naj-dobrite-ezici-za-programirane/>
- [5]<https://developer.android.com/studio/index.html>
- [6]<http://docs.oracle.com/en/>
- [7]<https://softuni.bg/blog/why-android-deserves-your-attention>
- [8]<https://infinum.co/the-capsized-eight/eclipse-is-dead-for-android-development-and-i-helped-kill-it>
- [9]<https://firebase.google.com/docs/>
- [10]<https://github.com/PhilJay/MPAndroidChart/wiki>
- [11]<https://dzone.com/articles/how-annotations-work-java>
- [12]<https://developers.google.com/maps/documentation/>
- [13]<http://stackoverflow.com/questions/29782808/what-does-fragmentmanager-and-fragmenttransaction-exactly-do>
- [14]<http://www.kircherelectronics.com/blog/>
- [15]<https://github.com/firebase/FirebaseUI-Android>

**Git Repository:** <https://github.com/Bzahov98/GodOfRowing>

## Съдържание:

<b>УВОД</b>	<b>3</b>
<b>ПЪРВА ГЛАВА.</b>	<b>5</b>
Проблематика	5
Съществуващи продукти	5
Endomondo	5
RowingInMotion (RiM)	5
Обобщение на съществуващите продукти:	6
Ендомондо	6
RowingInMotion	6
<b>ВТОРА ГЛАВА.</b>	<b>7</b>
<b>ИЗБОР НА ИЗПОЛЗВАНИТЕ ТЕХНОЛОГИИ</b>	<b>7</b>
Избиране на операционна система и програмен език	7
Избор на език	7
Компонентите на едно Android приложение са:	8
Други използвани термини:	9
Избор на средата за разработка(IDE)	9
Избиране на версия на Android	10
Избор на зависимости (dependencies) на проекта	11
Официални зависимости:	12
Зависимости на трети лица	12
FireBase Authentication	12
FireBase Realtime DataBase	12
Google Maps	13
MPandoridChart	13
Избор на Бекенд(BackEnd) – Firebase Database	14
Подготовка за свързване	15
Използвани сензори от Android устройството	17
Видове сензори, поддържани от ОС Android.	18
Използвани сензори:	19
Акселерометър	19

Сензор за линейно ускорение	20
Структура на проекта	20
Пакети на приложението	20
Пакет com.bzahov.elsys.godofrowing	20
Пакет com.bzahov.elsys.godofrowing.AuthenticationActivities	20
Пакет com.bzahov.elsys.godofrowing.Fragments	20
Пакет com.bzahov.elsys.godofrowing.RecyclerView	20
Пакет com.bzahov.elsys.godofrowing.ResultTabContent	21
Пакет .godofrowing.Models	21
ОПИСАНИЕ НА ИЗИСКВАНИЯТА КЪМ ПРОГРАМНИЯ ПРОДУКТ	21
Функционални изисквания	21
Поставяне на Android Устройството	22
Чрез самоделна установка	22
Чрез Rowing Boat Smartphone Mount	22
Описание на спомагателните класове за Firebase Database	23
Класът за потребителската информация (User)	23
Класът съдържащ информацията от тренировката	24
Реализиране На MyLocation class	24
<b>ТРЕТА ГЛАВА.</b>	<b>25</b>
Пакет com.bzahov.elsys.godofrowing.AuthenticationActivitie	25
Реализация формата за вписване (LogIn)	25
Реализация на форма за регистриране (SignIn)	27
Пакет com.bzahov.elsys.godofrowing;	29
Реализация на основния екран MainActivity	29
3.2.1.1 Абстрактният фрагмент за графика BaseChartFragment	31
Фрагмент за изчисляване на линейното ускорение	32
Регистриране на сензорите	33
Изчисляване на линейното ускорение	33
Засичане на нов загребок	35
Комуникацията между Фрагмент и Екран	37
Фрагментът с Google карта	38
Изпращане на дейността от тренировката	39

Екрана разпределител ResultActivity	40
Екранът за резултатите от тренировката	41
Екранът за историята на всички тренировки	42
<b>ЧЕТВЪРТА ГЛАВА.</b>	<b>43</b>
Изисквания	43
Инсталация	43
През електронна поща	43
През Google Play	44
Инструкции за използване	44
Влизане в профил	44
Чрез собствен Акаунт	44
Чрез анонимен Акаунт	45
Регистриране	45
Запознаване с основните параметри, които се отчитат в основния екран	46
Започване на записването на тренировката	47
Бутона за започване на тренировката Start Activity	47
Бутона за временно спиране Pause	47
Бутона за продължаване Resume	47
Бутона за прекратяване Stop	48
Резултати от текущата тренировка	48
Преглед на историята на тренировките	49
<b>ПЕТА ГЛАВА</b>	<b>50</b>
<b>Заключение</b>	<b>50</b>
Постижения в дипломната работа	50
Виждане за усъвършенстване на разработката	50
<b>Използвана Литература:</b>	<b>51</b>
<b>Съдържание:</b>	<b>52</b>