Portfolio - SmartMobile

iOS

Dimitar Matev - 27 June 2021

Duo App Project - Corona Monitor

After the specialisation kickoff I need to find a teammate for the first of two projects. The purpose of this project is to generate an app idea, validate it through different methodologies and carry out the implementation in an agile way. This way we will get an idea of what it takes to build an app from scratch and hopefully make it ready for Google Play.

Idea generation

To kickstart the project me and my partner Dimo decided to pick a brainstorming methodology so we can generate ideas in a structured way. This will increase the quality of ideas which means that we can find a good app idea faster.

After some researching we decided to combine two methodologies for brainstorming - **Rapid Ideation** and **Starbursting**. The purpose of the first technique is to generate as many ideas as possible in a short amount of time without any judgments during the process. After we are done with it we are gonna go through the ideas and criticise/approve them. By the end of this stage we should be having an idea of what the app will be for so we can proceed to the next methodology - **Starbursting**. We will be using it to look deeper into our chosen idea and potentially find problems surrounding it. For the purpose we will produce 6 questions which start with each one of the following -

"**Who** …", "**What**…", "**Where**…", "**When**", "**Why**" ,"**How**…". These keywords should inspire questions that sometimes need answering but nobody normally would ask.

Rapid Ideation

After our idea generation we looked into and criticised the following ideas:

- **Recommend a song based on your mood**
  - **(+/-)** Good idea but it might be too simple
- **Find lost items**
  - **(-)** Needs extra hardware to work. Does not seem feasible for the timeframe
- **Auto silence your phone while in a meeting**
  - **(-)** Possible problems with changing the setting on an iOS device
- **Stock market news and recommendations**
  - **(-)** Might be hard to do since none of us understands the finances that well
- **Corona monitor**
  - **(+)** There are not many existing apps on the app store that track the progress

- **(+)** Currently everyone is impatient for the lockdown to be over so the will be looking for extra information
- **(-)** After Corona is gone the app will be useless

Starbursting

In this stage we chose to dive deeper into the Corona monitor idea. That was the idea that made most sense to us since we are really looking forward of getting vaccinated and help end this pandemic. But since we currently don't have the opportunity to do so the other way is by informing the public on the matter. We see that in different countries different proportions of the population don't like vaccines or still don't believe that the Corona virus exists and that creates a problem. In the Netherlands the vaccine will be given with the priority on age which means if you are next in line and don't want a vaccine, a dose will be wasted and the process will slow down. By creating this app we hope to inform the people and spread awareness of the situation in an easy and convenient way.

Who is our target group? - Our target group is everyone that is interested in the corona pandemic. It's safe to assume that all of us have interest in it.

What information would they want to see? - the current rate of infections, total infections, people cured , deaths, and where we are with the vaccination.

When will be a good time to release it? - Hopefully we are seeing the end of the pandemic so we need to develop the app as soon as possible , otherwise we are risking having an app that won't matter anymore.

Where will be our application used? - We will first make sure to target our country (The Netherlands), but it should be possible to be used Worldwide.

Why would the users choose our application? - There are not many Covid tracking apps in the AppStore at the moment and none of them are tracking the vaccination progress which we believe is of the most interest currently.

How are we going to implement it? - We will have a back-end service which will spider news websites for up to date information and it will provide an API for our mobile app.

User personas

In the next stage I went ahead and started with the user personas for this project. To do so I talked with a couple of friends about the virus and their frustrations. After conducting the research I created the following personas which should accurately represent our target audience.

Name: Krasen Angelov

Age: 23 years old

Location: Eindhoven, The Netherlands

Language: Bulgarian, English

Family status: Single child

Stage of life: final year art student at university

Character: Friendly, Social

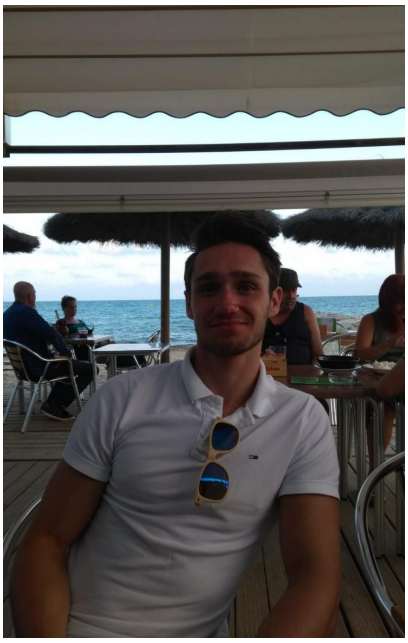
Motivations: Krasen is an ex-pat who hasn't seen his parents for more than a year now. He is getting really anxious about it because his grandfather died and he didn't have the chance to see him for the last time.

Goals: He wants to know when he will get vaccinated because that means he will be able to travel again back to his home country and see his family. He also wants to know if the vaccines are working.

Frustrations: His motivation to work and study for the university is at an all time low and he isn't really happy about it.

Professional background: Photography

Name: Luuk Joormann



Age: 20 years old

Location: Eindhoven, Netherlands

Language: Dutch, English

Family status: He has a little sister

Stage of life: Working, No university yet, Has a girlfriend

Character: Friendly, Hard-working, optimistic

Motivations: On the weekend Luuk Joormann is a really social guy. He always goes out with his friends. But because of corona he is not able to enjoy their company that much. He knows it's not responsible to do so.

Goals: He wants to know when he will get vaccinated because that means he will be able to see his friends more often and in bigger groups.

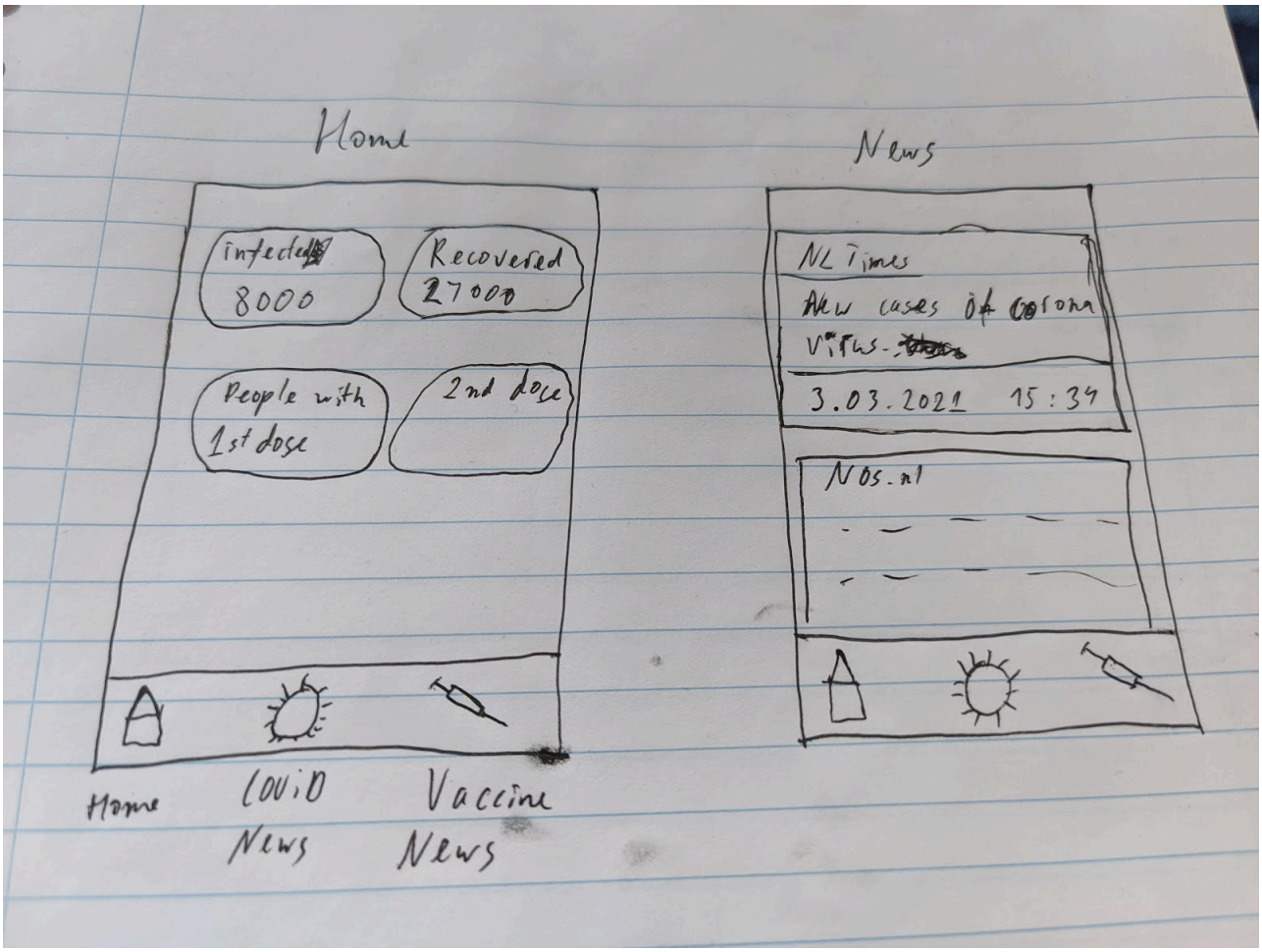
Frustrations: It's not so difficult to find information on the pandemic but it would be useful if everything he needs is in one place.

Professional background: Team leader of a pizza restaurant

Prototype

We have reached a point in this project where we have an idea, we validated it and we also got to know our target audience. The next step is to layout an initial design and see what feedback we can get from it. I chose to make a paper prototype to put my ideas into the physical world since it is pretty straightforward and fast method.

Paper prototype v1.0



In this first sketch you see two of the main screens that I imagined ur application having. The first one will be our home screen. It will be used to display general information about the current situation at the country that you are located. The information will be about the current number of infected people, the number of people that have recovered and the most important part - how the vaccination is going in the form of number of doses administered. The second and also the third screen will be about news surrounding the Corona virus. The information will be split into two sections hence

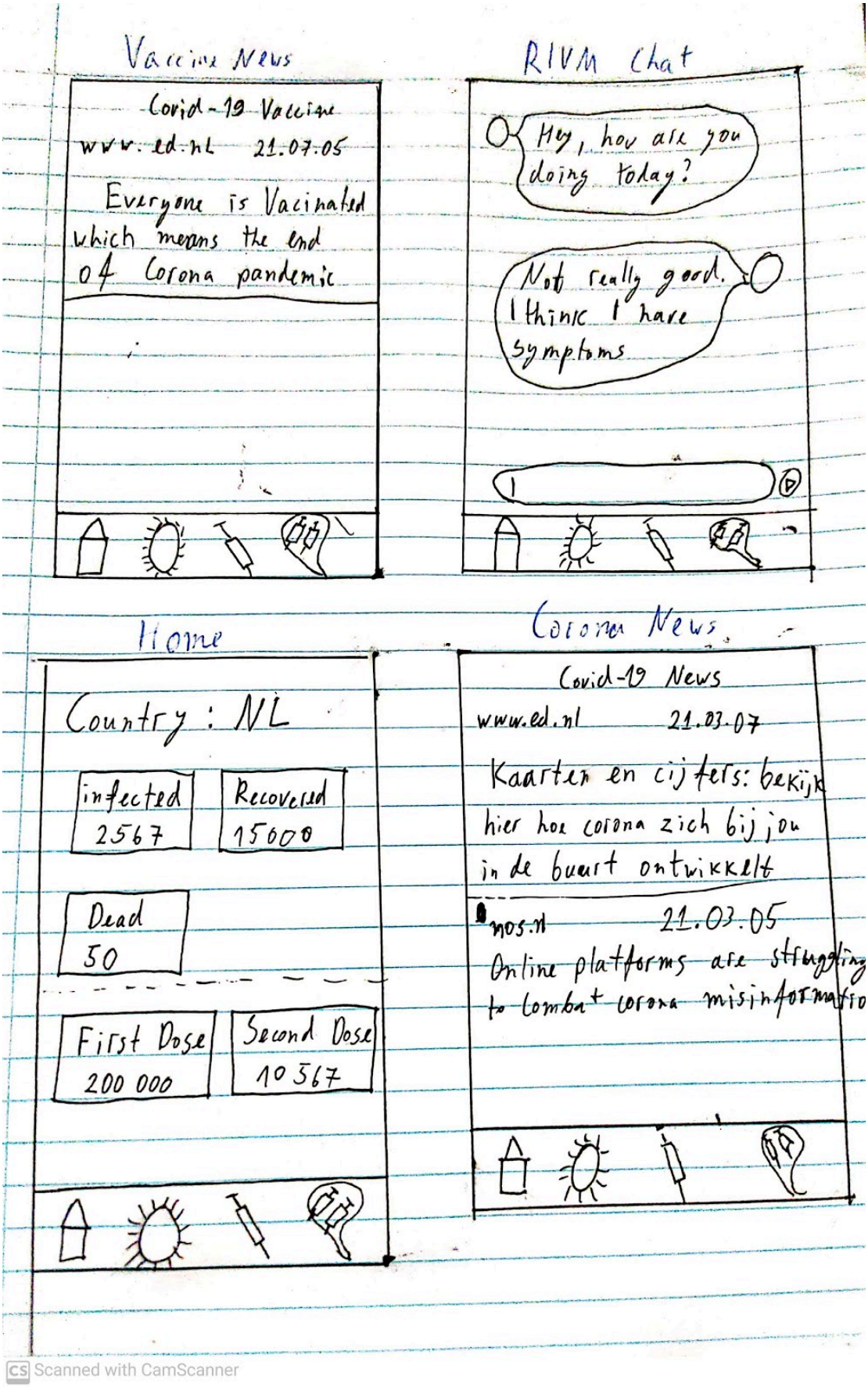
the two identical screens. One will be about the Covid news and the other one will be specific for the vaccination news.

After the sketch I started an interview process in which I wanted to see If I am going in the right direction. I went with my sketch to some representatives of the target audience that I know.

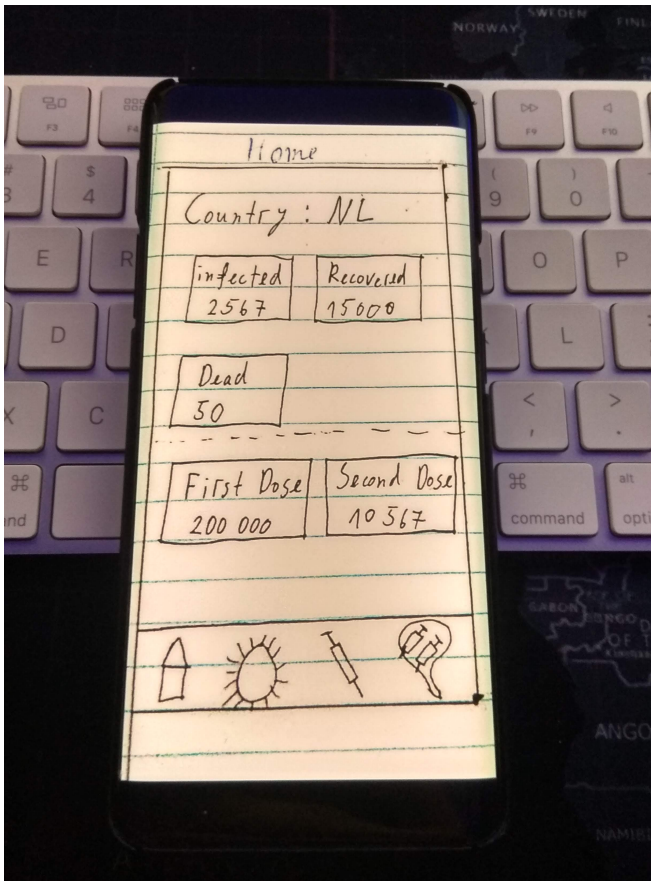
After I was finished with it the collection of feedback and suggestions I was really happy with the results. The first suggestion offered to me was to combine the two news sections into one, since my future users didn't see why they would want to have virus and vaccine news separate. I used the moment to ask if they would be more interested in seeing local and world Covid news instead? They agreed that it would be better since it would give them a good overview of how the situation is globally but most importantly they would be able to track the local situation with ease. This is especially valuable if they decide to travel to a foreign country.

With this data in mind I want ahead and created the second version of the paper prototype.

Paper prototype v2.0



In this version I've added a label on the home screen that would tell the user in which country he is currently located and the statistics for it. I also implemented the two separate news sections for local and world news as by the feedback collected. Finally I decided to add a 4th and final screen that would be a direct chat with RIVM. This would give the user direct connection with the authorities responsible for informing people on their questions and worries.



Next step is again to show it to my potential users and discuss the possible changes if they are needed. This time I decided to make it a clickable prototype with the help of an app called Marvel app which is available on Google play store (since my personal phone is an Android).

Since the new design was mostly what my users expected there wasn't much feedback on the home and news screens other than positive. They really liked the RIVM chat idea but expressed concerns that it would not happen since they don't currently support live chat on their site. This would mean that they don't have people ready to answer question by users on a timely manner. This was a valid point of which I didn't think of but I will definitely take into consideration.

Another feedback collected was from the teachers. They suggested an implementation of a Corona certificate for the people that are already vaccinated. This is also another valid suggestion into which we need to look into.

After doing some research on the Covid certificate me and Dimo concluded that It's not feasible for the deadlines which we have to complete this project. The reason for this is the lack of definitive information on when will the certificate will be announced, what format is it going to have and much more. This creates a level of uncertainty which we cannot afford to be in so we decided to replace the last (RIVM chat) with a vaccination poll. The idea behind it is that it will gather feedback from other users that are open to vaccination and want to share their experience and side effects. This way the more hesitant part of the population can make an informed and timely decision.

User requirements

At this point of the project me and Dimo had to clarify which features are most important and which we could leave for later implementation. To complete this task we decided to use the MoSCoW prioritisation method so everything is structured and we don't lose focus on not so valuable features.

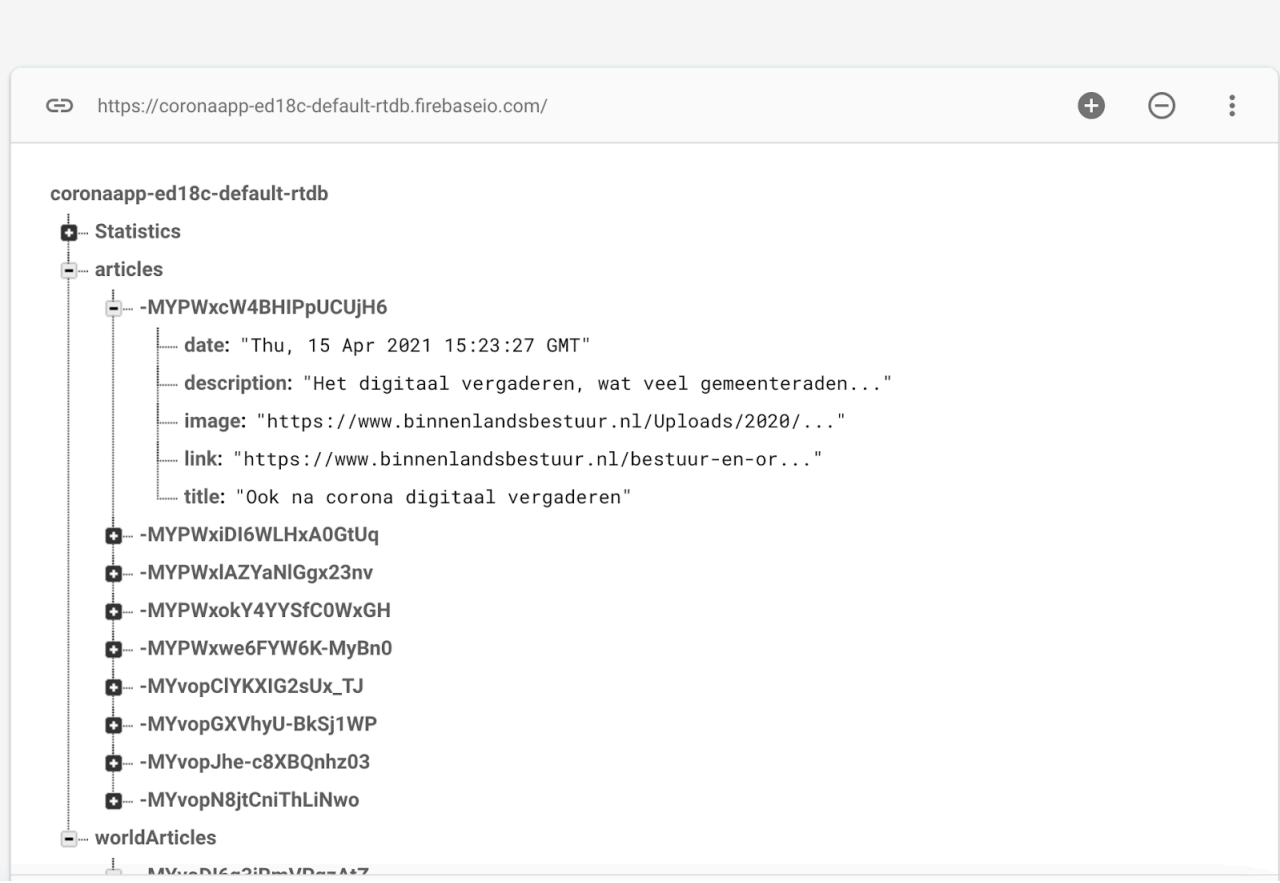
Requirement	Must have	Should have	Could have	Won't have
User can log into the app			X	
User can chat with RIVM				x
User can view Vaccine news				x
User can view local news for corona	x			
User can view global news for corona	x			
User can view local statistics compiled from the research		x		
User can give feedback for research	x			
User can view local corona numbers, based on GPS location	x			

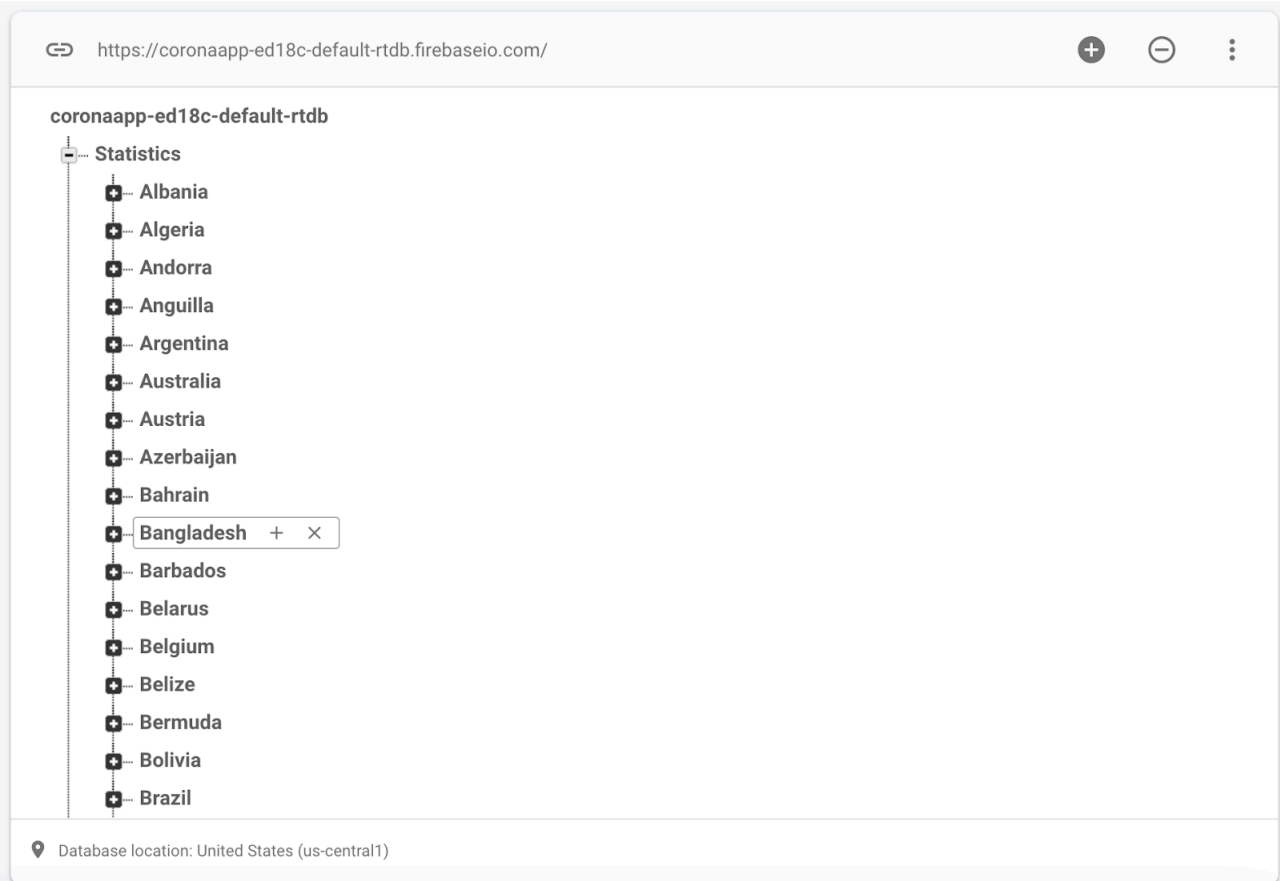
Implementation

In this stage of the project we as a team have to decide how the general architecture will look like so we can effectively start working on it. The main activities that our app has to perform are gathering information from the web and displaying it in a friendly and easy to use interface. Unfortunately it's not feasible to ask a smartphone to crawl the web for the information that it need, it's just not how it works. This will drain the 5G data of the user and it will kill the battery. Also it will be really slow to load which would make the app unusable. Because of this we decided to split the project into 3 parts - Back-end , Database and iOS application.

Database - Firebase

For the database we decided to use Firebase Realtime database which enables us to store our data in JSON format. This will help us store the data really easy since we don't need a complex relational database. Another benefit is the easy integration of Firebase in mobile apps which eliminates the need of creating and hosting an API service. We would only need to push the data to the database and the iOS app would be able to fetch and use it with no compatibility issues.





In the database we have 3 objects - “Statistics” , “articles” and “worldArticles”. “Statistics” will hold information for each and every country. This information will be used in the Home screen of the application. “worldArticles” will be a collection of articles from all over the world and “articles” are articles spidered for The Netherlands. When we later add support for more countries for example Bulgaria, a new object will be created in the database called “articlesBG” in which all the articles for that country will be located.

Back-end - Python and Scrapy framework

After doing some research on possible technologies and frameworks I've decided that Python will be best suited for the current situation. This is based on the fact that I am familiar with the technology from previous project that I have done. Another driving factor behind my decision is the huge number of frameworks and libraries available that can help me extract any data publicly accessible.

As mentioned above the main function of our back-end is to go through the news websites and collect data for articles that give information about coronavirus. This happens with the help of the so called spiders that we will create using the Scrapy framework. When the articles are scraped they will be automatically sent to the Firebase Realtime database.

```
import scrapy
import firebase_admin
from firebase_admin import db


class BinnenlandbestuurSpider(scrapy.Spider):
    name = 'Binnenlandbestuur'

    def start_requests(self):

        headers = {
            'authority': 'www.binnenlandsbestuur.nl',
            'cache-control': 'max-age=0',
            'sec-ch-ua': '"Google Chrome";v="89", "Chromium";v="89", ";Not A Brand";v="99"',
            'sec-ch-ua-mobile': '?0',
            'dnt': '1',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_3) AppleWebKit/537.36 (K
            'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
            'sec-fetch-site': 'same-origin',
            'sec-fetch-mode': 'navigate',
            'sec-fetch-user': '?1',
            'sec-fetch-dest': 'document',
            'accept-language': 'en-US,en;q=0.9,bg-BG;q=0.8,bg;q=0.7',
            'cookie': '_ga=GA1.3.1540676482.1616757285; _gid=GA1.3.1904432756.1616757285; accep
        }
        urls = [
            # 'https://www.binnenlandsbestuur.nl/coronavirus',
            'https://www.binnenlandsbestuur.nl/rss/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse, headers=headers)

    def parse(self, response):

        cred_obj = firebase_admin.credentials.Certificate('/Users/dimmat/Documents/Mitko/Fontys
        default_app = firebase_admin.initialize_app(cred_obj, {
            'databaseURL': 'https://coronaapp-ed18c-default-rtdb.firebaseio.com'
        })

        ref = db.reference("/articles")
```

In the above screenshot you can see one of the finished spiders. This spider is used to collect data from **binnenlandsbestuur.nl** and populate the database with articles for The Netherlands. To create the spider I had to make a new class which would inherit from scrapy.Spider. Then I need to implement two method - **start_requests** and **parse**. In the first one we specify some site specific variables like cookies, headers and/or payload data. We also need to find the url to an overview page that would give us all the articles we need. In the case of **binnenlandsbestuur.nl** they were really kind to leave an RSS feed which provides us with all of the information we need like - **<link> , <title>, <pubDate>, <description>** and **<image>**.

```
counter = 0
for description in response.xpath('//item/description[contains(text(),"corona") or contains(text(),"curfew")]/ancestor::item'):
    counter +=1
    ref.push({
        'title': description.xpath('title').get().replace('<title>', '').replace('</title>', ''),
        'date': description.xpath('pubDate').get().replace('<pubDate>', '').replace('</pubDate>', ''),
        'description': description.xpath('description').get().replace('<description>', '').replace('</description>', ''),
        'link': description.xpath('link').get().replace('<link>', '').replace('</link>', ''),
        'image': description.xpath('image').get().replace('<image>', '').replace('</image>', ''),
    })
    yield {
        'title': description.xpath('title').get().replace('<title>', '').replace('</title>', ''),
        'date': description.xpath('pubDate').get().replace('<pubDate>', '').replace('</pubDate>', ''),
        'description': description.xpath('description').get().replace('<description>', '').replace('</description>', ''),
        'link': description.xpath('link').get().replace('<link>', '').replace('</link>', ''),
        'image': description.xpath('image').get().replace('<image>', '').replace('</image>', ''),
        # 'image': description.xpath('image').get().replace('<image>','').replace('</image>',''),
    }
```

In the **parse** method is where the processing and storing of the spidered information happens. First we authenticate in and reference the database in order to use it. Then we need to write an expression that would get the articles and filter them so we only take the corona articles. This will happen with the help of **xPath**. This is a language that is used to describe the path to an html/xml element. Since the RSS page that we spider is an XML page and in genera all webpages use HTML, xPath will be used in all of the spiders. The expression that can be seen in the above image will look into the descriptions of all articles and will return articles that have the keywords “corona” or “curfew”. Then we do a **foreach** on all filtered articles and converts the information into something that would be usable in the application.

iOS - Swift

This part of the project represents the data scraped and stored from the web in an easy to understand and navigate interface. It was developed with the programming languages Swift and UIKit was used for the creation of the visual experience. This app not only shows the information but also gathers feedback from its users about their experience with the different corona vaccines.

While working on the application I manly focused on the Home Screen. It required getting GPS location of the user and retrieving the data from Firebase. To do so I had to use CLLocationManager which will give me the precise location of the user and CLGeocoder which will use the coordinates (**line 55-57 and 62**) to return me the country code of the current country (**line 76**). With it I can go to the database and request the country specific data (**line 80**).

```
49          var currentLoc: CLLocation!
50          var lat, lng: Double
51
52          if(locationManager.authorizationStatus == .authorizedWhenInUse ||
53              locationManager.authorizationStatus == .authorizedAlways) {
54
55              currentLoc = locationManager.location
56              lat = currentLoc.coordinate.latitude
57              lng = currentLoc.coordinate.longitude
58
59
60              // Create Location
61
62              let location = CLLocation(latitude: lat, longitude: lng)
63
64              // Geocode Location
65              geocoder.reverseGeocodeLocation(location, completionHandler: { [self](placemarks, error) -> Void in
66
67                  guard error == nil else {
68                      print("Reverse geocoder failed with error" + error!.localizedDescription)
69                      return
70                  }
71                  guard placemarks!.count > 0 else {
72                      print("Problem with the data received from geocoder")
73                      return
74                  }
75                  let pm = placemarks![0] as CLPlacemark
76                  self.CountryLabel.text = pm.country! as String
77                  //self.CountryLabel.sizeToFit()
78
79
80                  StatisticsManager.getAllStatistics(country: pm.country!)
81
82
83                  let vaccine: Vaccine = StatisticsManager.currentCountryVaccine
84
```

For this I have created a **StatisticsManager** which has the database reference. By using a manager class we are creating a separation between the Controller logic and the database. This way it will be easier to extend the functionality down the road.

```swift
import Foundation
import FirebaseDatabase
import SwiftyJSON

class StatisticsManager {
    let defaults = UserDefaults.standard

    static var ref: DatabaseReference!
    static var currentCountryVaccine: Vaccine = Vaccine()

    static func getAllStatistics(country:String) -> Void {
        self.ref = Database.database().reference()


        self.ref.child("Statistics").observeSingleEvent(of: .value, with: { snapshot in
            let dispatchGroup = DispatchGroup()
            for child in snapshot.children.allObjects as! [DataSnapshot] {
                dispatchGroup.enter()
                let vaccine = Vaccine.parseJSON(JSON(child.value ?? []), JSON(child.key))
                if vaccine.country.compare("US", options: .caseInsensitive) == .orderedSame {
                    vaccine.country = "United States"
                    print("Country set to UNITED STATES")
                }
                if vaccine.country.compare(country, options: .caseInsensitive) == .orderedSame {
                    self.currentCountryVaccine = vaccine
                }

            }


            dispatchGroup.leave()
            dispatchGroup.notify(queue: .main){

            }
        })
    }
}
```

Inside this class you can see that we have only one method **getAllStatistics** which accepts the country code as a parameter. After that it goest to the “Statistics” part in the database for the specific country and gets the data. In order for swift to parse the JSON data I also implemented a model class **Vaccine** in which has all the required parameters of the receiving data.

```swift
import Foundation
import SwiftyJSON

class Vaccine{

    var dosesAdministered: String
    var dosesPer: String
    var fullyVaccinated: String
    var vaccineModel: String
    var country: String
```

```
34      static func parseJSON(_ json: JSON,_ json1: JSON) -> Vaccine {
35          let vaccine = Vaccine()
36          if let dosesAdministered = json["doses_administered"].string{
37              vaccine.dosesAdministered = dosesAdministered
38              print(dosesAdministered)
39          }
40          if let dosesPer = json["doses_per_1000"].string {
41              vaccine.dosesPer = dosesPer
42              print(dosesPer)
43          }
44
45          if let fullyVaccinated = json["fully_vaccinated_population"].string {
46              vaccine.fullyVaccinated = fullyVaccinated
47              print(fullyVaccinated)
48          }
49          if let vaccineModel = json["vaccine_model"].string {
50              vaccine.vaccineModel = vaccineModel
51              print(vaccineModel)
52          }
53          if let country = json1.string {
54              vaccine.country = country
55              print(country)
56          }
57
58          return vaccine
59      }
```

Then we implement **parseJSON** method which will take the json response, get the parameters, store it into a new object of type Vaccine and then return it. Finally after getting the data the only thing left is to update the UI as follows:

```
80                  StatisticsManager.getAllStatistics(country: pm.country!)
81
82
83                  let vaccine: Vaccine = StatisticsManager.currentCountryVaccine
84
85
86
87                  print("DosesAdministeredLabel_____> " + vaccine.dosesAdministered)
88
89                  DosesAdministeredLabel.text = vaccine.dosesAdministered
90                  DosesPerLabel.text = vaccine.dosesPer
91                  FullyVaccinatedLabel.text = vaccine.fullyVaccinated
92                  VaccineTypeLabel.text = vaccine.vaccineModel
```

With this our Home Screen is complete.

Group Project - Music Running App

After the project kickoff meeting me and Dimo were placed in a team of 4 including us. The purpose of this project is to create a running app that would stimulate the user to run. This needs to be achieved through playing user specific music that stimulates him. For the purpose the client has implemented a machine learning algorithm that keeps track of the song preferences of the user the test data of which we requested. This is needed in order to make our app compatible with the data that the client is gathering and experiment with the sorting of the tracks.

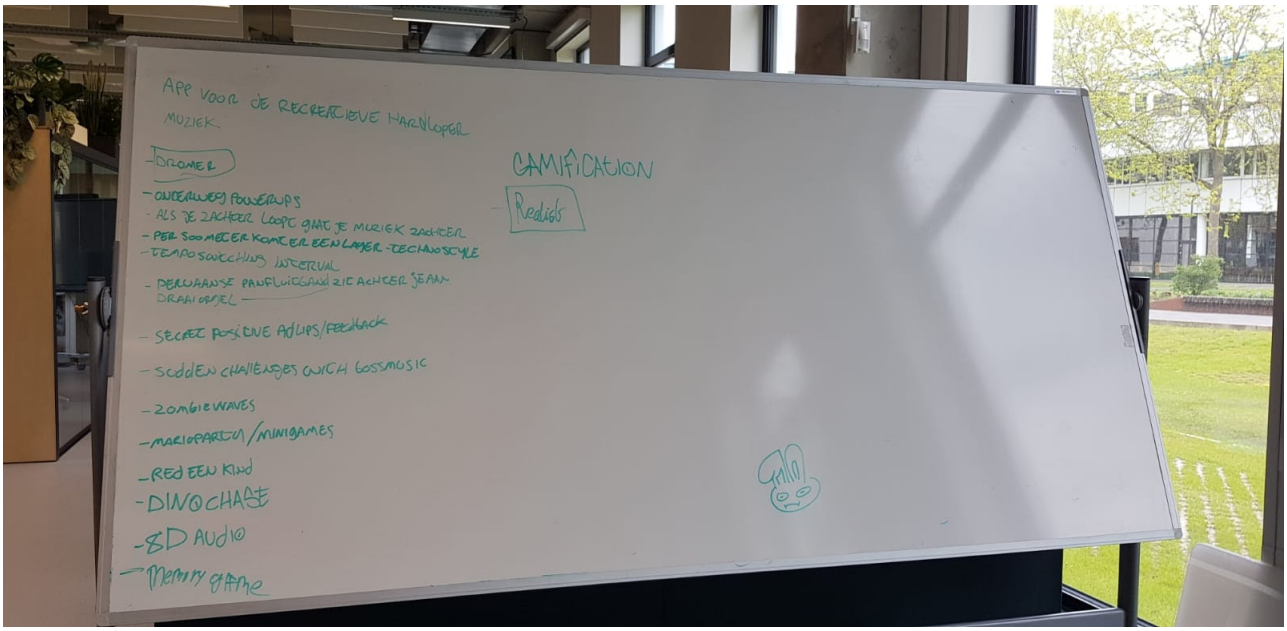
Idea generation

In this phase our group chose to use the Rapid ideation and Starbursting methodologies as per my recommendation for managing our ideas.

Rapid Ideation

- **Recommend a song based on your mood**
  - **(+/-)** Good idea but it might be too simple
- **Find Power-Ups on the way**
  - **(-)** Power-Ups don't have real life benefits.
- **Music reacts on your running speed**
  - **(-)** Only aiming for speed and not for health.
- **LayerBuilder where every … meters ran you build another layer of music**
  - **(-)** Might be hard to do since none of us understands the separation of layers in music that well
- **TempoSwitching Intervals**
  - **(-)** Tempo Switching will be hard to implement using real life obstacles.
- **Pan Flute band chasing you**
  - **(+/-)** Good idea but it might be too simple
- **Secret Positive adlibs/feedback**
  - **(-)** Might scare users
- **Sudden Challenges with Boss Music**
  - **(+/-)** Good idea but it might be too hard and challenges will be hard to implement
- **Zombie Waves**
  - **(-)** Users will run faster and not be more controlled and healthy.

- **Mario Party/Mini Games**
  - **(+/-)** Good idea but it might be too hard to implement minigames and multiplayer mode
- **Dino Chase**
  - **(+/-)** Good idea but might get boring fast or will scare users.
- **Using 8D audio to create a experience**
  - **(+/-)** Good idea, unfortunately not enough time to discover the possibilities and use of 8D sound combined with making an application.
- **Memory Game**
  - **(+)** Good idea for training mind and body at the same time.



Starbursting

In this stage we have chosen to stick with the Running Program Application. This application will give the user a running program combined with music that fits the tempo and goal of the program part.

Who is our target group?

Our target group is based on fun runners, who run on the regular but not on the professional level. These people go out running with each their own reason.

Therefore the focus will be specified on creating an environment where every user has the opportunity to enjoy their running session.

What information would they want to see?

Our target group wants to have the feeling of achieving goals, which can be losing weight, getting condition or even clearing the mind.

Therefore we will only supply positive-minded feedback towards the user. This will make them feel better about themselves, not depending on the performance of the user. This doesn't mean we won't give any feedback related to someone's performance. The feedback based on performance will come in data, like basic running stats, such as average speed, total distance of session and possibly how many calories were burned during the session.

When will be a good time to release it?

It would be awesome to finish the project before the summer of 2021, because people can finally go out again and want to do activities outside. Also our deadline will be on 24 of June so that comes in perfect.

Where will be our application used?

Our application will be used in the Netherlands at first. When the application has been finished and validated the next step would be to go international with the application.

Why would the users choose our application?

The difference between the usual running application and our application is that the audio feedback is the most important in the application. The preset programs combined with your favorite music and running program will give the user structure and reduce stress which leads to the best session experience.

How are we going to implement it?

We will read out the API setup by our PO. This data will be used to select the right music at each point of time. This combined with the reactive running program will guide the user through the running session, taking away the stress of having to make decisions in tempo, songs, and breaks.

User personas

In the next stage I went ahead and started with the user personas for this project. To do so I again talked with a couple of friends but this time about the their running habits and frustrations. After conducting the research I created the following personas which should accurately represent our target audience.

Name: Krasen Angelov

Age: 23 years old

Location: Eindhoven, The Netherlands

Language: Bulgarian, English

Family status: Single child

Stage of life: final year art student at university

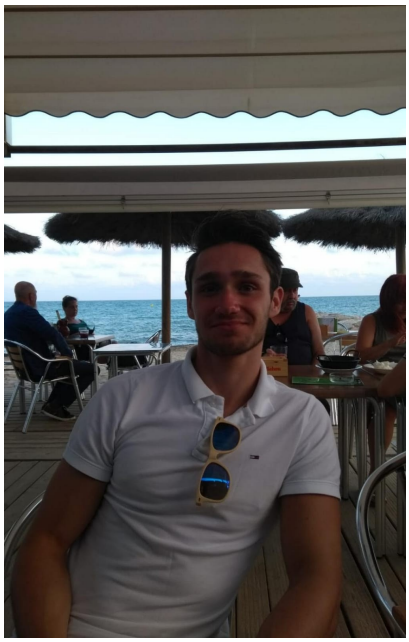
Character: Friendly, Social

Motivations: To release stress, or to reward himself for a few days of long study sessions, Krasen Angelov heads out to the nearest park. which hosts a couple of abandoned buildings and a skateboard playground.

Goals: He wants to change something in his training routine to make the experience more interesting.

Frustrations: His motivation to work and study for the university is at an all time low and he isn’t really happy about it. The monotonous running at the same place over and over again doesn’t help either but there aren't any other places around.

Professional background: Photography



Name: Luuk Joormann

Age: 20 years old

Location: Eindhoven, Netherlands

Language: Dutch, English

Family status: He has a little sister

Stage of life: Working, No university yet, Has a girlfriend

Character: Friendly, Hard-working, optimistic

Motivations: On the weekend Luuk Joormann really likes to take his car and drive around in order to search for some nice places to run. He is in it for the new tracks and hiking in the area.

Goals: He would like to discover new places in the area on his own to relieve himself from stress and the normal world.

Frustrations: It's always difficult for him to find a new track every now and then.

Professional background: Team leader of a pizza restaurant

App Description

In this part of the project we are finalising the ideation phase. This is done by creating a complete summary of what our app is supposed to do.

"So how long has it been since you went out practising sport or just taking a walk or run? In the time we sat at home, our body got a lot less movement than before. As sport and movement has been hard to practise during the last year now it's more important than ever to get people to exercise.

Pace will guide you through your running session as a whole other form of experience. So will the application guide you through your session based on the selected program. Your favourite music will be played and switched up to achieve the optimal running session.

With the use of music and a running program the user will be led through the session, including breaks, sprints, warming up and warm down. Therefore the user doesn't have to stress about the program, and can totally focus on him/herself exercising.

The user will receive positive feedback, which is not dependent on the performance of the user.

Users will be able to have insights into their achievements, statistics and personal information. This will include average speed, total distance, time used, music favourites and program completion."

User requirements

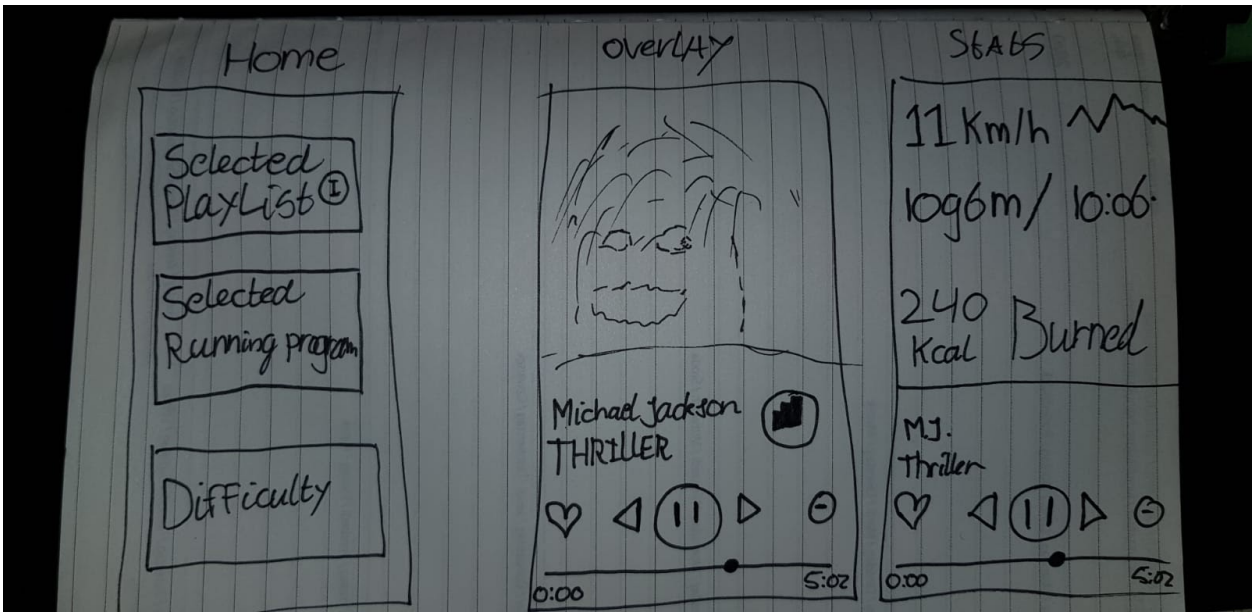
At this point of the project me and the group had to clarify which features are most important and which we could leave for later implementation. To complete this task we decided to use the MoSCoW prioritisation method so everything is structured and we don't lose focus on not so valuable features.

Requirement	Must have	Should have	Could have	Won't have
User can select own playlist	X			
User can select multiple programs		X		
User can view daily statistics	X			
User can view session statistics	X			
User can select difficulty		X		
User can log in into application			X	
User can like/dislike songs		X		
User has map insight of session			X	
User will be able to select a pre-planned route				X
User will be able to select terrain dependent running program				X

Prototype

We have reached a point in this project where we have an idea, we validated it and we also got to know our target audience. The next step is to layout an initial design and see what feedback we can get from it. Our teammates chose to make a paper prototype to put their layout ideas into the physical world.

Paper prototype v1.0

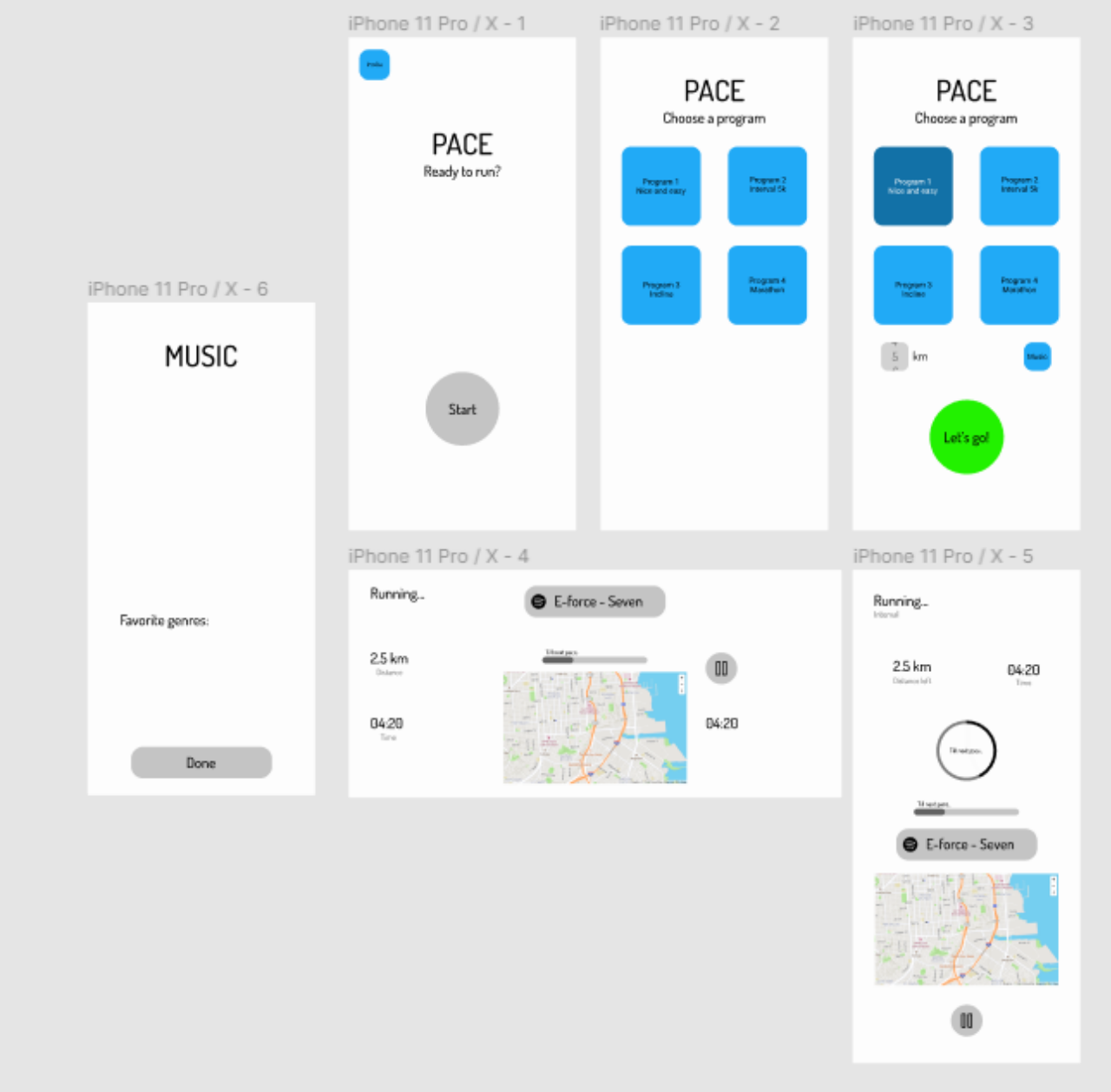


Our initial sketch shows 2 screens. The main screen will offer 3 options to the user. The first one is to select a playlist for running, then a training program depending on which the music will be sorted and lastly the difficulty of the session. The second screen will be the statistics screen. The user will see it only if he has started his running session. It will provide media controls for the music and statistics for the current running session such as calories burned, distance ran and the time since the practice started.

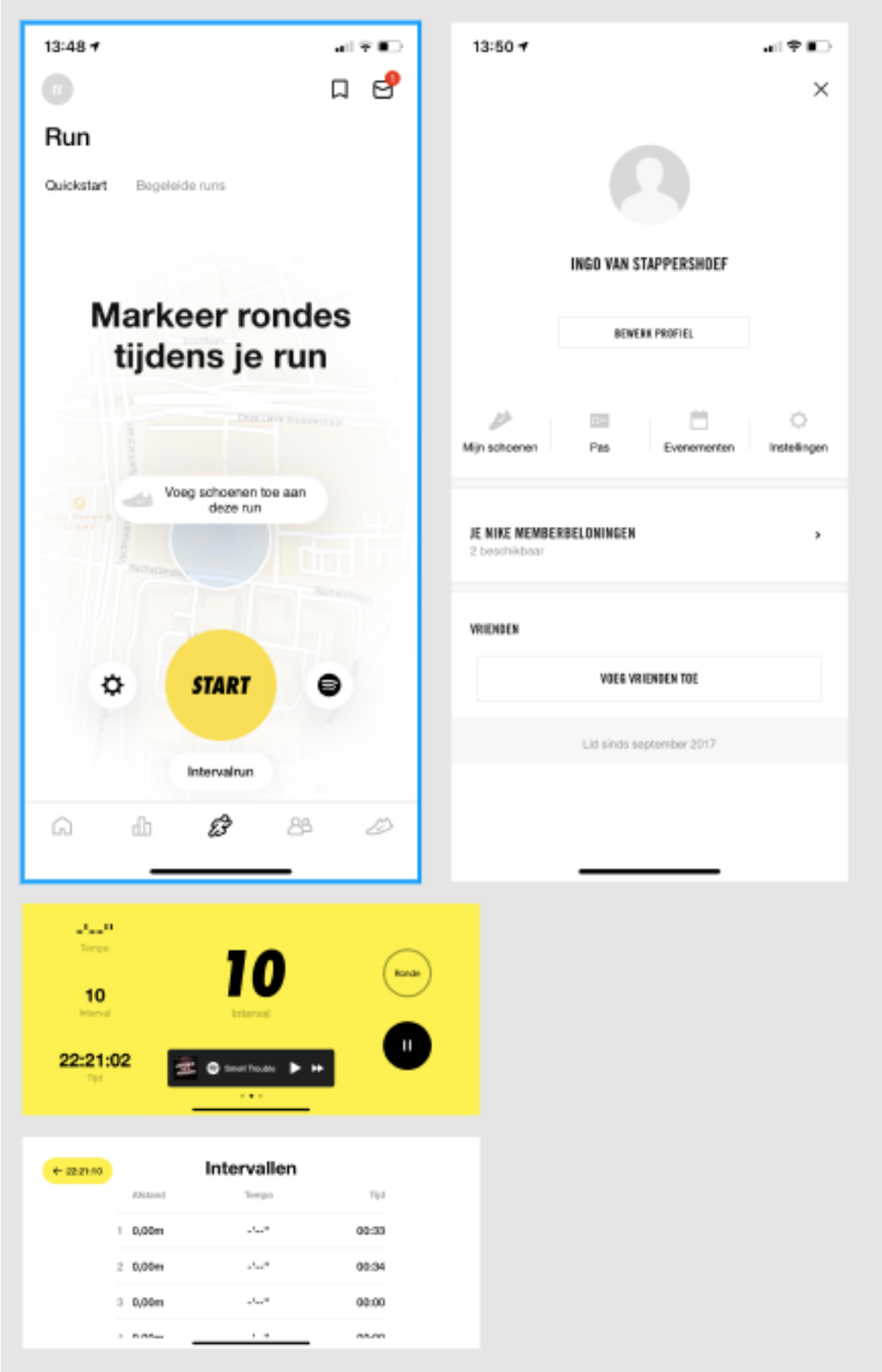
Design

In this part of the project our media teammates went forward and created our application design.

Initial design

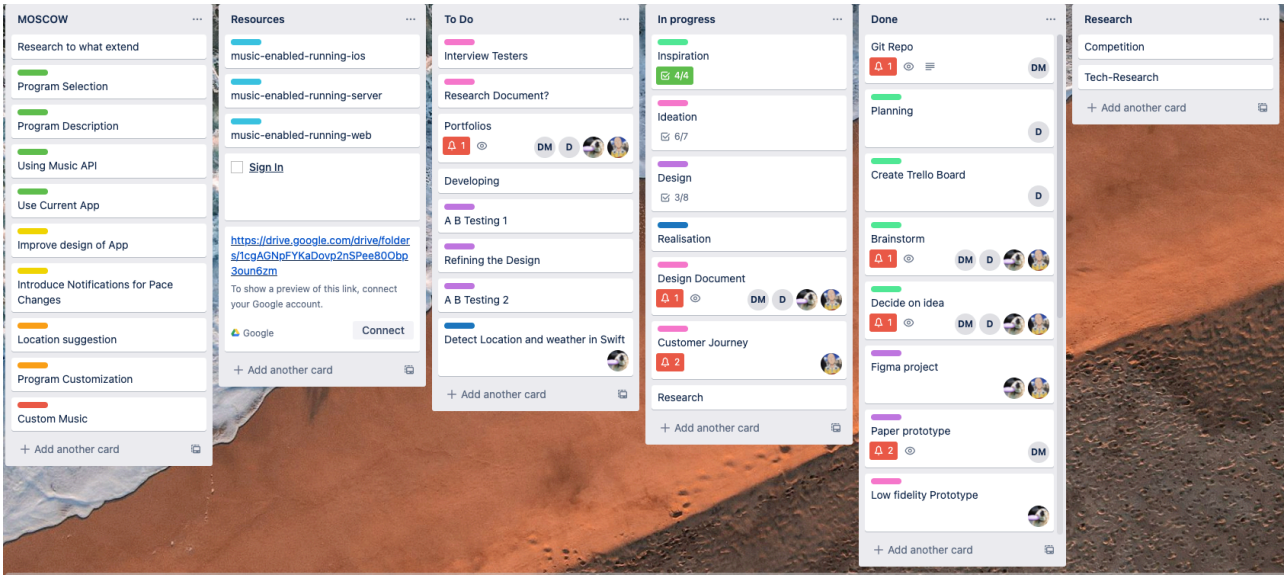


Improved design

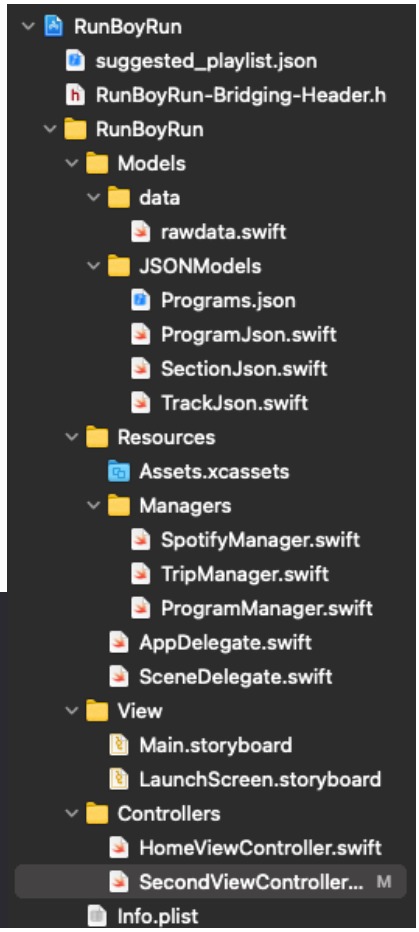


Implementation

In this stage of the project me and Dimo had to look into the list of tasks and start with their implementation. To make it easier for us, Dimo created a trello board in which he transferred all of the tasks from the MoSCoW list and added some more for the other guys as well.



In the picture on the right you can see the general outline of the project. Let's start by looking into our 3 models that we have. They are used in a similar way as in our Corona app, the only difference being that the data will be loaded from a file. This is because we will store our different training programs in JSON file. By doing it like that we can easily add new programs later on without changing the logic of the application. We also load data for the songs of the user. All of this is possible because of the **Codable** type alias. It takes care of the code that encodes and decodes the data from a serialised format.




```
9  struct TrackJSONElement: Codable {
10     let acousticness: Double
11     let analysisSections: [[String: Double]]
12     let analysisURL: String
13     let artist: String
14     let artistGenres: [String]
15     let coverURL: String
16     let danceability: Double
```

The next step is to implement the methods for reading the file, and for parsing the data.

```
private func readLocalFile(forName name: String) -> Data? {
    do {
        if let bundlePath = Bundle.main.path(forResource: name,
                                             ofType: "json"),
            let jsonData = try String(contentsOfFile: bundlePath).data(using: .utf8) {
            return jsonData
        }
    } catch {
        print(error)
    }

    return nil
}

private func parse(jsonData: Data) -> [TrackJSONElement]{
    var decodedData: [TrackJSONElement] = []
    do {
        decodedData = try JSONDecoder().decode([TrackJSONElement].self,
                                               from: jsonData)

        print("==================================")
    } catch {
        print("decode error")
    }
    return decodedData
}
```

Now that we have all the data we need its time to implement a manager class which will use to generate a playlist of songs that match the training program loaded. In our case it's called **TripManager**. Inside we have 4 arrays of songs which should represent our 4 different types of sections that we would have in a training program. These arrays start empty and later on are getting populated. If we combine them into one big array we shouldn't have any duplicates.

```
10  class TripManager{
11      var tracks: TrackJSON
12      //let program: ProgramJSON
13      var inteseTracks: [TrackJSON]
14      var joggingTracks: [TrackJSON]
15      var walkingTracks: [TrackJSON]
16      var relaxTracks: [TrackJSON]
17
18      var highQ: TrackJSONElement?
19      var lowQ: TrackJSONElement?
20      var middleQ: Double
21
22      var highToMidQ: Double
23      var MidToLowQ: Double
24
25      var highTempo: TrackJSONElement?
26      var lowTempo: TrackJSONElement?
27      var middleTempo: Double
28
29      var highToMidTempo: Double
30      var MidToLowTempo: Double
31
32      var totalSongs: Int
33      var songsPerTypeOfSection: Int
```

```
self.highQ = tracks.max {$0.q < $1.q}
self.lowQ = tracks.min{$0.q > $1.q}
self.middleQ = (highQ!.q+lowQ!.q)/2

self.highToMidQ = (highQ!.q - middleQ)/2
self.MidToLowQ = (middleQ - lowQ!.q)/2

self.highTempo = tracks.max {$0.tempo < $1.tempo}
self.lowTempo = tracks.min {$0.tempo > $1.tempo}
self.middleTempo = (highTempo!.q+lowTempo!.q)/2

self.highToMidTempo = (highTempo!.q - middleTempo)/2
self.MidToLowTempo = (middleTempo - lowTempo!.q)/2
```

In this section we define values which will help us get a better understanding of the playlist that we are working on. Here are the variables and what they do:

highQ - will have the track with the highest Q
lowQ - will have the track with the lowest Q
middleQ - will take the average Q

highToMidQ - return the offset from high Q to middle Q
MidToLowQ - return the offset from middle Q to low Q

highTempo - will have the track with the highest Tempo
lowTempo - will have the track with the lowest Tempo
middleTempo - will take the average Tempo

highToMidTempo - return the offset from high Tempo to middle Tempo
MidToLowTempo - return the offset from middle Tempo to low Tempo

To better understand them we need to look into how we use them for example in **getIntenseSongs()**.

```
//HIGH BPM HIGH Q
func getIntenseSongs() -> [TrackJSONElement]{


    tracks.sort{$0.tempo < $1.tempo && $0.q < middleQ && $1.q < middleQ && $0.tempo < middleTempo && $1.tempo < middleTempo} // Ascending
    //so the high bpm and q are at the end
    var trackList: [TrackJSONElement] = []
    print("----------------INTENSE-SONGS-----------------")
    for _ in 0...songsPerTypeOfSection{

        if let unwrappedTrack = tracks.popLast(){  // here we are poping the last element
            trackList.append(unwrappedTrack)
            print("---" + unwrappedTrack.artist)
            print("---" + String(unwrappedTrack.tempo))
            print("---" + String(unwrappedTrack.q))
            print("-------------------------------------")
        }else{
            print("ERROR in SORTING the TRACKS")
        }

    }

    //for track in trackList
    print("----------------INTENSE-SONGS-----------------")
    return trackList
}
```

In this method we take the tracks list which contains the training data and information about all of the songs the user likes to listen to. Then we use the **sort()** method and inside of it as parameter we place our sorting logic. In this case we sort the songs first by tempo so the songs with the highest tempo are last on the list. After that we sort by Q by comparing it with **middleQ**. This way we make sure that both of the songs that we compare are with good Q which means that the user will be motivated to run. Finally we ensure that the tempo is also above the middle point which ensures that we will have energetic music.

After the sorting is done we should have our best songs at the end of the array. This way we can easily take songs and remove them from our playlist. The reason behind it is to not have any duplicates while playing the songs.

Reflection

I personally enjoyed the Smart Mobile specialisation. It gave me the opportunity to dive into iOS development for the first time. But this course wasn't only learning the theory behind developing an app, it was more than that. The specialisation provided me with an opportunity to try first do a project in a really small group. That made me learn a lot because I had to participate in every aspect of the project - from the ideation phase all the way to making the documents/research and implementing it. But after the second project, I learned that there is no better way to develop an application than when you are doing it with a group of teammates that share your ideas and even hobbies.