



## Produktrapport

# Svendeprøve

**Elev**

Mathias Frederik Græsholt

**Projektnavn**

FunRun

**Uddannelsessted**

TECHCOLLEGE

Struervej 70,

9220 Aalborg

**Elevplads**

EMD International

Niels Jernes Vej 10,

9220 Aalborg Ø

**Projektperiode**

21. marts 2022 - 19. april 2022

**Afleveringsdato**

19. april 2022

**Fremlæggelsesdato**

26. april 2022

**Vejledere**

Frank Rosbak

Lærke Brandhøj Kristensen

**Censor**

Allan Kaa Jensen

## Forord

Denne rapport er en af to skrevet til svendeprøveprojektet FunRun af Mathias Frederik Græsholt.

Projektet er udarbejdet i perioden 21. marts 2022 til 19. april 2022, med vejledning fra Frank Rosbak og Lærke Brandhøj Kristensen.

Denne rapport omhandler en proof-of-concept løsning på en motionsapp.

Systemet gør en bruger i stand til at optage løbeture og anden motionsbevægelse, at se alle deres tidligere ture på en overskuelig måde, og at få detaljeret information om en given tur, deriblandt visning på et kort.

Rapporten indeholder kravspecifikation og testkonditioner, samt hvordan projektet lever op til de obligatoriske og valgfrie projektkrav der er stillet. De teknologier der indgår i projektet beskrives, og der gives et indblik i funktionaliteten bag systemets hovedfunktion, at optage og vise ture. Der er ydermere inkluderet brugervejledninger i hvordan systemet benyttes.

Jeg takker alle involverede i projektforløbet.

# Indholdsfortegnelse

Læsevejledning .....	5
Indledning.....	5
Kravspecifikation .....	6
Testkonditioner .....	7
Krav .....	8
Information om teknologier .....	10
API.....	10
Entity Framework .....	10
JSON .....	10
bcrypt.....	10
Database.....	11
T-SQL.....	11
PWA .....	12
Vue.js .....	12
Hosting.....	12
Overordnet arkitektur .....	13
PWA - Afsender .....	13
API - Modtager.....	15
Database.....	17
API - Afsender .....	17
PWA - Modtager .....	18
Resultat.....	22
Brugervejledninger .....	23
1. Installation af app .....	23
2. Oprettelse af bruger .....	23
3. Log ind .....	24
4. Ny løbetur .....	24
5. Mine løbeture .....	25
6. Vis løbetur.....	25
7. Slet løbetur .....	26
Kildeliste .....	27

## Læsevejledning

Denne rapport er en af to der hører til svendeprøveprojektet FunRun.

I produktrapporten beskrives projektets produkt og teknologier. I procesrapporten beskrives forløbet, og hvordan produktet blev formet.

Det anbefales at man starter med at læse produktrapporten, og får et indblik i hvad projektet er, før man læser procesrapporten.

Samlet kildeliste findes bagerst i rapporten.

I denne rapport bruges en del tekniske forkortelser, men især to er vigtige at kende:

**API** står for Application Programming Interface, og er en type program der faciliterer kommunikation mellem to andre teknologier.

**PWA** står for Progressive Web App, og er en type hjemmeside der kan downloades som en app.

Begge teknologier bliver beskrevet dybere i rapportens teknologiafsnit.

Projektets kode, dele af hvilken forekommer i afsnittet Overordnet Arkitektur, er dokumenteret på engelsk idet resten af koden er skrevet på engelsk.

Links til projektets filer i denne rapport, er afkortet med tjenesten Bitly. Dette er gjort for læselighedens skyld.

## Indledning

Som teknologien skrider frem, kræves mindre og mindre af vores kroppe, som flere timer end aldrig før får lov at sidde stille.

Procenten af befolkningen der lider af overvægt, er stadigt stigende, og dette er et problem for folkesundheden der nedsætter livskvaliteten.

Folk der ønsker at komme dette til livs, kan vælge et væld af motionsmuligheder for at holde sig i form, men en af de mest tilgængelige motionsformer er nok løb.

Løb kræver ikke noget udstyr, og næsten alle kan gøre det på et niveau der passer til dem.

Et brugbart udstyr til løb kunne dog være et system der kunne vise ens ture og give en indblik i relevant statistik deromkring, og derved fastholde løbere, som uden konkret bogføring og visualisering af deres fysiske motion hurtigt kunne miste interesse.

Projektet FunRun er et svar på dette, og denne rapport omhandler produktet og teknologierne bag det.

## Kravspekifikation

Selve produktet består af et API med tilhørende database og en PWA.

Denne tabel viser kravene til systemets forskellige dele.

Krav ID	Kategori	Type	Krav
SO1	Server	Opsætning	Database skal være opsat med tabeller til at indeholde brugere, ture, og punkter.
SF1	Server	Funktionalitet	API skal kunne modtage data fra PWA og lagre denne i databasen.
SF2	Server	Funktionalitet	API skal kunne sende data fra databasen til PWA.
SF3	Server	Funktionalitet	API skal udregne relevante statistikker når brugeren efterspørger data.
SS1	Server	Sikkerhed	Databasen skal opbevare sensitive brugeroplysninger sikkert.
PO1	PWA	Opsætning	PWA skal kunne tilgås og downloades eksternt.
PF1	PWA	Funktionalitet	PWA skal kunne registrere nye brugere.
PF2	PWA	Funktionalitet	PWA skal kunne logge brugere ind og ud.
PF3	PWA	Funktionalitet	PWA skal kunne sende brugeres lokation til API efter ønske.
PG1	PWA	GUI	PWA skal kunne præsentere tidligere løbeture for brugere.
PG2	PWA	GUI	PWA skal kunne præsentere brugbar information om gennemsnitshastigheder, m.m. for brugere.

## Testkonditioner

Denne tabel viser de testkonditioner der bestemmer om kravene er opfyldt af det endelige produkt.

Test ID	Opfyldt	Til dels opfyldt	Ikke opfyldt	Test Case
SO1	X			Databasetabeller skal inspiceres, og kolonnens datatyper kontrolleres, i Microsoft SQL Server Management Studio efter opsætning.
SF1-A	X			Kontrollér via Swagger og Microsoft SQL Server Management Studio at API'ets endpoints modtager og lagrer data korrekt.
SF1-B	X			Kontrollér i Microsoft SQL Server Management Studio at data fra PWA forekommer i databasen som forventet.
SF2-A	X			Kontrollér via Swagger at API'ets endpoints returnerer data som forventet.
SF2-B	X			Kontrollér at data fra databasen forekommer i PWA'en som forventet.
SS1-A	X			Kontrollér via Microsoft SQL Server Management Studio at brugers kodeord bliver hashed i databasen.
SS2-B	X			Kontrollér at brugere oprettet med samme kodeord får forskellige hashes i databasen.
PO1	X			Kontrollér at en smartphone kan downloade og benytte PWA'en.
PF1	X			Kontrollér at brugere kan oprettes uden problemer.
PF2	X			Kontrollér at brugere kan logge ind og ud uden problemer.
PF3	X			Kontrollér at oplysninger fra PWA'en om position kommer ordentligt igennem til databasen og med en acceptabel frekvens.
PG1	X			Kontrollér at PWA'en kan fremvise tidligere ture korrekt og hensigtsmæssigt.
PG2	X			Kontrollér at PWA'en kan give nyttig information og statistik til brugeren.

## Krav

Dette afsnit beskriver hvordan de faglige krav der er stillet til projektet er opfyldt.

### Obligatoriske elementer

Projektet har fem obligatoriske faglige krav som er beskrevet nedenfor.

#### Konfigurationsstyring

Git bliver benyttet til konfigurationsstyring af projektet.

Projektets GitHub repository indeholder både begge større dele af projektet, API og PWA, samt rapporter og tidsplaner m.m.

Projektets GitHub repository kan findes på: <https://github.com/Graesholt/Svendeprøve>

Derudover er GitHub også konfigureret med GitHub Actions til at bygge, teste, og deploye API, og bygge og deploye PWA ved hvert push. Det vil sige at API er sat op med fuldbåret Continuous Integration, og PWA er sat op med Continuous Deployment.

Projektets GitHub Actions side kan findes på: <https://github.com/Graesholt/Svendeprøve/actions>

#### Sikkerhed

Der er blevet taget højde for sikkerhed i projektet ved at brugers kodeord bliver hashed i databasen ved brug af en 'langsom' algoritme, og aldrig bliver opbevaret i plain text.

Når en bruger er logget ind, bliver de tildelt en JSON Web Token, som senere bruges til at krydstjekke at de er hvem de giver sig ud for, når de beder om information. Det er ikke muligt for en bruger at ændre på denne token uden at gøre den ugyldig.

Yderligere sker al udveksling af information mellem API og PWA via HTTPS, den krypterede udgave af HTTP.

Systemet validerer også brugernavn og kodeord. Det vil sige al tekst i systemet der er indtastet af en bruger, valideres før denne interagerer med databasen på nogen måde. På denne måde opnås sikkerhed mod SQL injection angreb.

#### Test

Projektets API har en række unit tests der sikrer at metoden til at validere brugerdata fungerer som forventet. Fordi denne del af API'et egner sig til unit testing, giver dette mulighed for at bruge test driven development. Da brugernavns- og kodeordsvalideringen implementeres, skrives testene som det første, for at have en række scenarier der tillades eller forbydes, og derefter bliver koden skrevet.

API'ets unit tests kan findes på: <https://bit.ly/GraesholtFunRunApiUnitTests>

Ud over dette, testes API'ets endpoints. Dette sker ved black-box tests, da API'ets afhængighed af tokens gør endpoint metoder vanskelige at unit teste. Testene kalder i stedet endpoints udefra og



kontrollerer at de får de korrekte responses, og at uautoriserede brugere ikke kan tilgå eller manipulere information der ikke tilhører dem.

API'ets black-box tests kan findes på: <https://bit.ly/GraesholtFunRunApiBlackboxTests>

Til PWA er en række brugertestcases vedlagt som bilag. Brugertests er brugt i stedet for autotests i dette projekt, da udviklerens erfaring med autotest viser at autotestcases er utroligt smarte og effektive, men tager lang tid at opbygge korrekt.

Brugertestcases er vedlagt som bilag 1.

Ud over dette har systemet været benyttet løbende af et antal uafhængige brugere.

### **Database**

Projektets API har en tilhørende database, sat op via Entity Framework, og Code-First princippet. Denne database omfatter data i form af brugere, løbeture, og punkter.

### **Server**

Begge projektets dele, samt dets SQL server, kører på Microsoft Azure. Hoveddelene, API og PWA kører som services, og er tilgængelige udefra.

En traditionel, fysisk server er trods forespørgsel ikke blevet stillet til rådighed for projektet.

### **Valgfri elementer**

Udover de fem obligatoriske faglige krav, skal projektet opfylde mindst et af en række valgfrie krav. Dette projekt opfylder to.

#### **Klient/Server**

Projektet har en server, i form af en SQL server, der kører på Microsoft Azure.

Projektets klient er PWA'en, som kan tilgå data fra serveren via API'et.

#### **App Udvikling**

Projektet besvarer kravet for App Udvikling ved dets brug af en Progressive Web App. Denne app er udviklet i Vue.js og kan køre på et udvalg af platforme. Appen tilgår information fra enhedens GPS, og er den del af systemet som brugere vil benytte til at interagere med API'et og databasen.

Link til app: <https://bit.ly/GraesholtFunRun>

# Information om teknologier

Dette afsnit omhandler information om de teknologier der indgår i dette produkt.

## API

Projektet benytter et API der er skrevet i ASP.NET. API står for Application Programming Interface, og et bindeled der tillader flere forskellige teknologier at udveksle data<sup>1</sup>.

Dette projekts API er bindeleddet mellem databasen og PWA'en. Dette tillader de to services at køre på separate enheder med forbindelse til internettet, frem for at skulle køre på samme enhed.

## Entity Framework

Projektets API benytter Entity Framework til at kommunikere med databasen.

Entity Framework er et ORM framework, som står for Object Relational Mapping. Det betyder at i stedet for at arbejde med data i tabeller, med kolonner og rækker, kan data tilføjes og tilgås som objekter. Dette gør dataene nemmere at arbejde med og forstå i koden<sup>2</sup>.

## JSON

Når data skal sendes mellem PWA og API sker dette i formatet JSON.

JSON står for JavaScript Object Notation, og er et dataudvekslingssprog der er nemt for mennesker at læse og forstå, hvilket gør det håndgribeligt at arbejde med. JSON er sproguafhængigt, men bruger konventioner som ligner C-Sprogfamilien, deriblandt C# som API'et er skrevet i<sup>3</sup>.

Det at JSON udveksler data i objekter gør det velegnet til udveksling af data med et API som er objektbaseret.

## bcrypt

Til beskyttelse af data, i form af hashing af kodeord, bruges bcrypt.

bcrypt har to store fordele over andre algoritmer når det kommer til kodeord hashing. For det første er bcrypt, modsat mange andre, ikke programmeret for hastighed og ydeevne, men med sikkerhed i højsædet. Når man hasher et kodeord, kan man vælge at hashe resultatet igen for at fordoble tiden et brute-force angreb skal investere i at knække det. Dette vil en bruger i teorien ikke mærke noget til, da man kan hashe et kodeord mange tusind gange på et enkelt sekund, men en brute-force svindler vil være tvunget til at hashe et større antal gange for hvert forsøg. Mange algoritmer gør dette, men bcrypt har som egenskab at den kan skalere sit antal af iterationer med computerens kraft, så man ikke ender med en masse gamle, sårbare kodeord, og en masse nye, sikrere kodeord. For det andet tager bcrypt et såkaldt salt, som bliver hashed sammen med kodeordet. Dette sikrer at folk med samme kodeord ikke vil have samme

<sup>1</sup> <https://aws.amazon.com/what-is/api/>

<sup>2</sup> <https://www.entityframeworktutorial.net/what-is-entityframework.aspx>

<sup>3</sup> <https://www.json.org/json-en.html>

hash i databasen, så en brute-force svindler vil skulle knække hvert kodeord separat, frem for at kunne tage en masse på samme tid, hvis det er blevet aflæst på forhånd at en mængde brugeres hashede kodeord er ens<sup>4</sup>.

## Database

Projektet benytter sig af en MS-SQL database sat op via Code-First princippet gennem Entity Framework. Dette betyder at man har bedt API'et om at generere en database ud fra en beskrivelse, i modsætning til at sætte en database op først og kode den sammen med API'et manuelt. Dette mindsker risikoen for manuelle opsætningsfejl, og gør også at databasen hurtigt kan sættes op igen, skulle dette være nødvendigt.

MS-SQL er et relationelt databasesystem, hvilket vil sige at man kan have data der har relationer til anden data. Disse relationer kommer i to typer; en-til-mange relationer, og mange-til-mange relationer. For eksempel, i dette projekt ser vi brugere, ture, og punkter. En bruger består af information vigtig for en bruger, for eksempel brugernavn og kodeord. En tur består af information om en given tur, for eksempel hvornår den foregår. Et punkt består af koordinater for hvor på kloden punktet befinder sig. Alle tre objekttyper har også Id'er, som man kan bruge til at lave relationer mellem dem. En tur skal, for eksempel, indeholde hvilken bruger der har været ude på turen, så hver tur refererer til et brugerId. Punkter er ikke meget værd hvis de ikke fortæller hvilken tur de hører til, så hvert punkt refererer til et turId. Disse relationer er begge eksempler på en-til-mange relationer. En mange-til-mange relation kunne for eksempel være hvis man byggede systemet så flere brugere kunne deltage i en given tur. I dette tilfælde ville man have en mellemtabel til at indeholde relationsdataene, som begge tabeller så ville have en-til-mange-relationer til. I den forstand kan en mange-til-mange relation siges at være en kombination af to en-til-mange relationer.

Den primære kommunikationsvej til en MS-SQL database er T-SQL.

### T-SQL

T-SQL bygger på SQL, Structured Query Language, som er standarden for at kommunikere med relationelle databasesystemer<sup>5</sup>.

SQL er deklarativt, modsat det meste andet programmering som er imperativt. Det vil sige at hvor man normalt fortæller et stykke software præcis hvad det skal udføre, kommando for kommando, vil man i stedet beskrive det resultat man ønsker, og så vil softwaren sørge for resten<sup>6</sup>.

---

<sup>4</sup> <https://auth0.com/blog/hashing-in-action-understanding-bcrypt/>

<sup>5</sup> <https://www.tutorialspoint.com/sql/sql-overview.htm>

<sup>6</sup> <https://vegibit.com/the-declarative-nature-of-sql/>

## PWA

En PWA, eller Progressive Web App, er en nem måde at udvikle en app der kan køre cross-platform, altså som virker både på Android og iOS, og sågar også en desktop PC. En PWA kodes i HTML, CSS, og JavaScript, så hvis man er komfortabel med de sprog i forvejen, kan man nemt gå til det.

En PWA får på en måde det bedste fra to verdener; den er tilgængelig overalt som en traditionel web-app, og den kan tilgå en enheds komponenter som en native app kan. En PWA kan dog ikke tilgå alle de komponenter en native app kan, men den tilgængelige funktionalitet er voksende<sup>7</sup>.

Dette projekts PWA er skrevet i et framework der hedder Vue.js.

## Vue.js

Vue.js er et library til at bygge reaktive webinterfaces<sup>8</sup>.

Vue.js er som udgangspunkt bygget på arkitekturmønstret MVVM, eller Model-view-viewmodel. Dette vil sige at interfacet (view) og logikken (model) er adskilt og snakker sammen via en mellemmand (viewmodel). Dette foregår via Databinding som Vue.js har indbygget, og som gør at man kan binde ui-elementer til bestemte data, som så opdaterer når dataene gør, uden at det behøver at være en del af logikken<sup>9</sup>.

Ydermere er Vue.js komponentbaseret, som vil sige at man kan definere forskellige komponenter, som for eksempel en navigations- eller side-bar, og så importere og bruge dem på flere sider. Det er noget der virkelig giver mening i forhold til andre former for programmering, hvor logik bør samles så meget som muligt, så der kun er et sted det skal ændres hvis det skulle blive aktuelt<sup>10</sup>.

## Hosting

Alle projektets dele er hostede på Microsoft Azure.

Microsoft Azure er Microsofts Cloud Computing platform, på hvilken man kan hoste en bred vifte af forskellige cloud services til at opbevare eller transformere data. Microsoft Azure tilbyder en pay-per-use model, og er den hurtigst voksende Cloud Computing platform på markedet<sup>11</sup>.

Projektets API er hosted i form af en App Service forbundet med en SQL database.

Projektets PWA er hosted i form af en Static Web App.

---

<sup>7</sup> <https://web.dev/what-are-pwas/>

<sup>8</sup> <https://v1.vuejs.org/guide/overview.html>

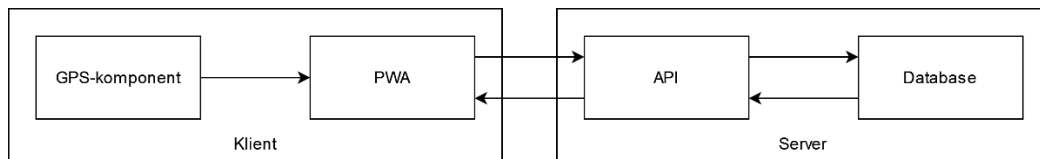
<sup>9</sup> <https://v1.vuejs.org/guide/overview.html>

<sup>10</sup> <https://v1.vuejs.org/guide/overview.html>

<sup>11</sup> <https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure>

## Overordnet arkitektur

Produktet består af tre dele; en PWA, en database, og et API der fungerer som bindeled derimellem.



Arkitekturdiagram tegnet via Draw.io

Dette afsnit vil forklare vejen som data tager gennem projektet, når man benytter projektets hovedfunktion, som er at optage ture, og se dem efterfølgende.

### PWA - Afsender

Når en bruger er klar til at løbe og trykker på 'Start Løbetur' bliver den følgende kode afviklet i PWA'en.

----- CODE START -----

```
await axios
  .post(process.env.VUE_APP_API_URL + "api/Run", "", {
    headers: { Authorization: `Bearer ${localStorage.getItem("jwtToken")}` },
  })
  .then(async function (response) {
    console.log(response.data);
    runId = response.data.runId;
    console.log(runId);
    //Ready to log points, now that runId is known
    status.value = "run started";

    //Test if wakeLock exists. Will fail on many traditional computers, but not on
    //devices such as phones.
    if ("wakeLock" in navigator) {
      try {
        //Set wakeLock, preventing screen from locking
        lock = await navigator.wakeLock.request("screen");
      } catch (err) {
        //Error occurred
        console.log("Wake Lock error: ", err);
      }
    }
  });
```

----- CODE END -----

(Fuld kode: <https://bit.ly/GraesholtFunRunPwaNewrun>)

Denne kode sender besked til API'et om at den gerne vil starte en ny tur, og med som header sender den sin token, så API'et kan kontrollere at forespørgslen er lovlig. Dette kald besvares med det objekt der er

oprettet i databasen og `runId` gemmes i en variabel der skal bruges når der tilføjes punkter, for at vide hvilken tur de hører til.

Ud over dette ændres `status.value` til senere brug, og der startes en `wakeLock`, som gør at en enheds skærm ikke låses automatisk.

Selve turens punkter bliver tilføjet til databasen med følgende kode.

```
----- CODE START -----
watchId = navigator.geolocation.watchPosition(
  (position) => {
    console.log("position", position);
    //Configure point object
    let point = {
      longitude: position.coords.longitude,
      latitude: position.coords.latitude,
      altitude: position.coords.altitude,
    };
    var pointLatLng = new L.LatLng(point.latitude, point.longitude);
    console.log("point", point);
    //If a run is in progress
    if (status.value == "run started" || status.value == "running") {
      //Post point to API
      axios.post(process.env.VUE_APP_API_URL + "api/Point/" + runId, point, {
        headers: { Authorization: `Bearer ${localStorage.getItem("jwtToken")}` } });
      //Add point to map polyline
      runPolyline.addLatLng(pointLatLng);
      //If first point since run started
      if (status.value == "run started") {
        status.value = "running";
        //Set timer start time
        startTime = Date.now();
        //Update timer ten times a second
        timerInterval = setInterval(updateTimer, 10);
        //Change button to show it will now end run
        refStartRunButton.value = "Stop løbetur";
      }
    }
  },
  () => {},
  { enableHighAccuracy: true }
);
----- CODE END -----
```

(Fuld kode: <https://bit.ly/GraesholtFunRunPwaNewrun>)

`navigator.geolocation.watchPosition()` bliver kørt sammen med oprettelsen af kortet når siden indlæses, og funktionen i den kører hver gang den får en ny position fra enhedens GPS. En enheds GPS kan optage mange forskellige data, men for projektet her er det kun væsentligt at behandle `latitude`, `longitude`, og `altitude`.

Så snart `status.value` er ændret til `"run started"` af det forrige kald, vil efterfølgende punkter, udover at opdatere kortets centerposition, blive sendt til databasen sammen med `runId` på den igangværende tur, samt tilføjes til `runPolyline`, som er en linje på kortet der viser ens bevægelse løbende.

Ud over dette startes en timer på siden, og knappen opdateres til at kunne stoppe turen. Når brugerens tur er slut, trykker de på knappen igen. Dette ender `watchPosition()` (via `clearWatch()`), ender timeren, og frigiver `wakeLock`, så der ikke sendes flere punkter, der ikke går ressourcer til at opdatere en timer der ikke længere er på skærmen, og brugerens enhed igen automatisk kan låse skærmen.

## API - Modtager

Når API'et modtager at der skal startes en ny tur, modtager den faktisk ikke anden data end brugerens token.

```
----- CODE START -----  
[HttpPost]  
public async Task<ActionResult<Run>> NewRun()  
{  
    var run = new Run();  
  
    //Do not trust client time, server time is reliable  
    run.dateTime = DateTime.UtcNow;  
    run.user = await _context.Users.FirstOrDefaultAsync(u => u.userId ==  
        GetUserId());  
    run.deleted = false;  
  
    _context.Runs.Add(run);  
    await _context.SaveChangesAsync();  
  
    return CreatedAtAction("GetRun", new { id = run.runId }, run);  
}  
----- CODE END -----
```

(Fuld kode: <https://bit.ly/GraesholtFunRunApiRunController>)

API'et bruger metoden `GetUserId()` til at læse `userId` ud fra den token der modtages i headeren, og bruger denne info til at finde det `User` objekt som turen skal tilhøre. `dateTime` genereres serverside i UTC-format for at tider bliver så ensartede og korrekte som muligt, frem for at bruge tiden fra en brugers enhed.

Når API'et modtager et **Point**, modtages punktets data, samt hvilket **Run** det hører til.

```
----- CODE START -----
[HttpPost("{runId}")]
public async Task<ActionResult<Point>> NewPoint(Point point, int runId)
{
    //Do not trust client time, server time is reliable
    point.dateTime = DateTime.UtcNow;
    var run = _context.Runs.Include("user").Include("points").FirstOrDefault(r =>
        r.runId == runId);

    if (run == null)
    {
        return NotFound();
    }

    if (run.user.userId != GetUserId())
    {
        return Unauthorized();
    }

    run.points.Add(point);
    //_context.Points.Add(point);
    await _context.SaveChangesAsync();

    return StatusCode(201);
}
----- CODE END -----
```

(Fuld kode: <https://bit.ly/GraesholtFunRunApiPointController>)

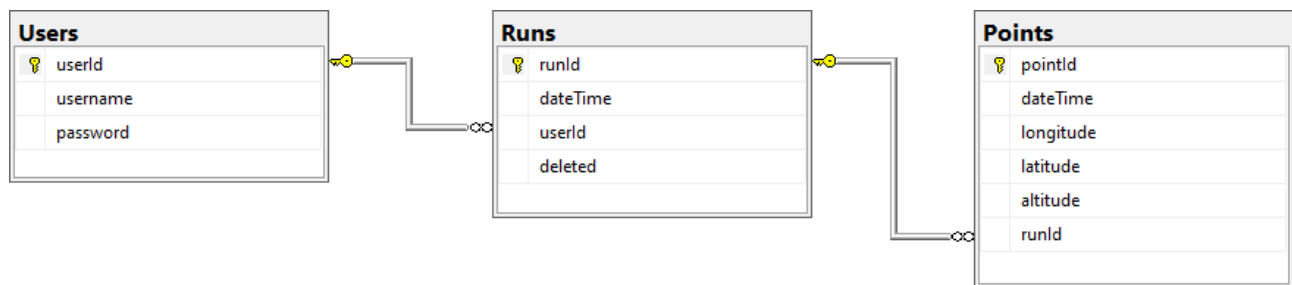
Punktet tilføjes til **run.points** i stedet for **\_context.Points** fordi systemet aldrig arbejder med **Points** fra databasen isoleret, men altid i konteksten af hvilket **Run** det tilhører. Derfor, ved at tilføje dem på denne måde, kan man bede om at få **Points** med som en del af et **Run** objekt, i stedet for at lave et separat databasekald for at få dem leveret efterfølgende. API'ets **PointController** har derfor ikke et endpoint til at udlevere **Points**, kun til at modtage dem.

I dette tilfælde bruges brugerens token kun til at sikre at kaldet er gyldigt, for at forhindre en forkert **User** i at tilføje ugyldige **Points** til **Runs** de ikke ejer. Der er ikke grund til at tilknytte et **Point** en **User**, da punktet i forvejen er tilknyttet et **Run**, som allerede er tilknyttet en **User**.



## Database

Projektets database er struktureret som påvist på entity relation diagrammet herunder.



Entity Relation Diagram (ERD) genereret via Microsoft SQL Server Management Studio 18

*Users* tabellen indeholder data om brugere, som for eksempel brugernavn og kodeord. Kodeord er hashede med bcrypt og aldrig opbevaret eller behandlet af systemet i plain text. Tabellen har også et `userId` som systemet bruger, men som brugeren aldrig ser.

*Runs* tabellen indeholder det data der er vigtigt for en tur, alt sammen, som forklaret ovenfor, genereret serverside når den bliver oprettet.

*Points* tabellen indeholder data vigtig for et enkelt punkt tilknyttet en tur.

## API - Afsender

Når en bruger har afsluttet en tur, bliver de viderestillet til en side der viser information om den. Samme side kan tilgås ved at vælge en tidligere løbetur på oversigten der vises efter login.

----- CODE START -----

```
[HttpGet("{runId}")]
public async Task<ActionResult<RunStats>> GetRun(int runId)
{
    var run = await _context.Runs.Include("user").Include("points")
        .FirstOrDefaultAsync(r => r.runId == runId);
    //Include("points") gets a list of points associated with this run as a
    //property on the object.

    if (run == null)
    {
        return NotFound();
    }

    if (run.user.userId != GetUserId())
    {
        return Unauthorized();
    }
}
```

```
if (run.points.Count() == 0)
{
    return UnprocessableEntity();
}

RunStats runStats = new RunStats(run);

return runStats;
}
----- CODE END -----
```

(Fuld kode: <https://bit.ly/GraesholtFunRunApiRunController>)

I stedet for at sende et Run objekt, som er hvad der er oprettet og lagret i databasen, sendes her et **RunStats** objekt. **RunStats** er et DTO, et Data Transfer Object, som ikke er lagret i databasen. I stedet bliver **RunStats** objektet genereret når brugeren beder om information om en tur. De nye properties der bliver beregnet ud fra de allerede tilgængelige data er turens distance (distance), turens varighed (duration), hvor hurtigt der blev løbet i gennemsnit (avgSpeedInMetersPerSecond), og hvor hurtigt der i gennemsnit blev løbet minut for minut (avgSpeedPerMinuteInMetersPerSecond). Disse data udregnes serverside for at spare brugerens enhed.

## PWA - Modtager

Når PWA'en modtager informationen bliver den formateret på siden og præsenteret for brugeren.

```
----- CODE START -----
<div class="center-div top-stat-div">
  <Card class="center-text stat-card">
    <template #title>
      <p class="stat-card-field">
        {{ refRun.duration }}<span class="subscript">{{ refDurationMilliseconds }}</span>
      </p>
    </template>
    <template #content>
      <p class="stat-card-field">Varighed</p>
    </template>
  </Card>

  <Card class="center-text stat-card">
    <template #title>
      <p class="stat-card-field">
        {{ refRun.distance }}<span class="subscript">{{ refDistanceUnit }}</span>
      </p>
    </template>
    <template #content>
```

```
        <p class="stat-card-field">Distance</p>
    </template>
</Card>
</div>

<div class="center-div bottom-stat-div">
    <Card class="center-text stat-card">
        <template #title>
            <p class="stat-card-field">{{ refRun.avgSpeedInMetersPerSecond }}<span
                class="subscript">km/t</span></p>
        </template>
        <template #content>
            <p class="stat-card-field">Gns. hastighed</p>
        </template>
    </Card>
</div>

<div id="run-map"></div>

<p class="center-text chart-header">Højdekurve</p>
<div class="center-div scheme-div">
    <line-chart :data="refAltitudePointList" xtitle="Tid" ytitle="Højde i m"
        class="scheme" empty="Henter data..." :curve="false" :points="false"
        :min="refAltitudeSchemeMin" :max="refAltitudeSchemeMax"></line-chart>
</div>

<p class="center-text chart-header">Gennemsnitshastighed pr. minut</p>
<div class="center-div chart-div">
    <line-chart :data="refAvgSpeedPerMinutePointList" xtitle="Minut" ytitle="Km/t"
        class="chart" empty="Henter data..." :curve="false"
        :points="AvgSpeedPerMinuteChartPoints" :min="refAvgSpeedPerMinuteChartMin"
        :max="refAvgSpeedPerMinuteChartMax" :xmin="refAvgSpeedPerMinuteChartStart"
        :xmax="refAvgSpeedPerMinuteChartEnd" :colors="['#ff6600']"></line-chart>
</div>
```

----- CODE END -----

(Fuld kode: <https://bit.ly/GraesholtFunRunPwaRun>)

Øverst i HTML'en ses tre `Card` elementer der vil indeholde varighed, distance, og gennemsnitshastighed. Efter disse følger en `div` der vil blive til sidens kort, og til sidst ses to `line-chart` elementer der viser turens højdekurve, og hastigheden per minut.

Alle disse elementer, undtagen kortet, bruger Vue.js' indbyggede databinding til automatisk at opdatere når deres databundne `ref()`-værdier ændrer sig.

----- CODE START -----

```
axios
    .get(process.env.VUE_APP_API_URL + "api/Run/" +
```

```
router.currentRoute.value.params.runId,
{ headers: { Authorization: `Bearer ${localStorage.getItem("jwtToken")}` },
})
.then(function (response) {
  console.log(response.data);
  refRun.value = response.data;
  //Header configured. UTC time converted to locale time with en-GB formatting
  refHeader.value = new Date(refRun.value.dateTime + "Z").toLocaleDateString(
    "en-GB") + " - " + new Date(refRun.value.dateTime + "Z").toLocaleTimeString(
    "en-GB");

  //Map: creation
  var map = L.map("run-map");
  //Map: tilelayer configuration
  L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
    attribution: 'Map data &copy; <a
      href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
      contributors',
    maxZoom: 19,
    tileSize: 256,
  }).addTo(map);

  //mapLatLngBounds holds the bounds we are interested in viewing on map
  var mapLatLngBounds = new L.LatLngBounds();
  //mapPointList holds the LatLngs of each point of the Run
  var mapPointList = [];
  //Loops through points configuring both variables
  refRun.value.points.forEach((element) => {
    var pointLatLng = new L.LatLng(element.latitude, element.longitude);
    mapLatLngBounds.extend(pointLatLng);
    mapPointList.push(pointLatLng);
  });

  //Set map bounds
  map.fitBounds(mapLatLngBounds);

  //Map: polyline configuration
  var runPolyline = new L.Polyline(mapPointList, {
    color: "blue",
    opacity: 0.5,
    smoothFactor: 1.5,
  }).addTo(map);

  //Map: start tooltip configuration
  var startTooltip = L.tooltip({
    direction: "top",
    permanent: true,
```

```
    })
    .setLatLng(mapPointList[0])
    .setContent("Start")
    .addTo(map);

//Map: end tooltip configuration
var endTooltip = L.tooltip({
  direction: "bottom",
  permanent: true,
})
.setLatLng(mapPointList[mapPointList.length - 1])
.setContent("Slut")
.addTo(map);

[CODE OMITTED]
})
----- CODE END -----

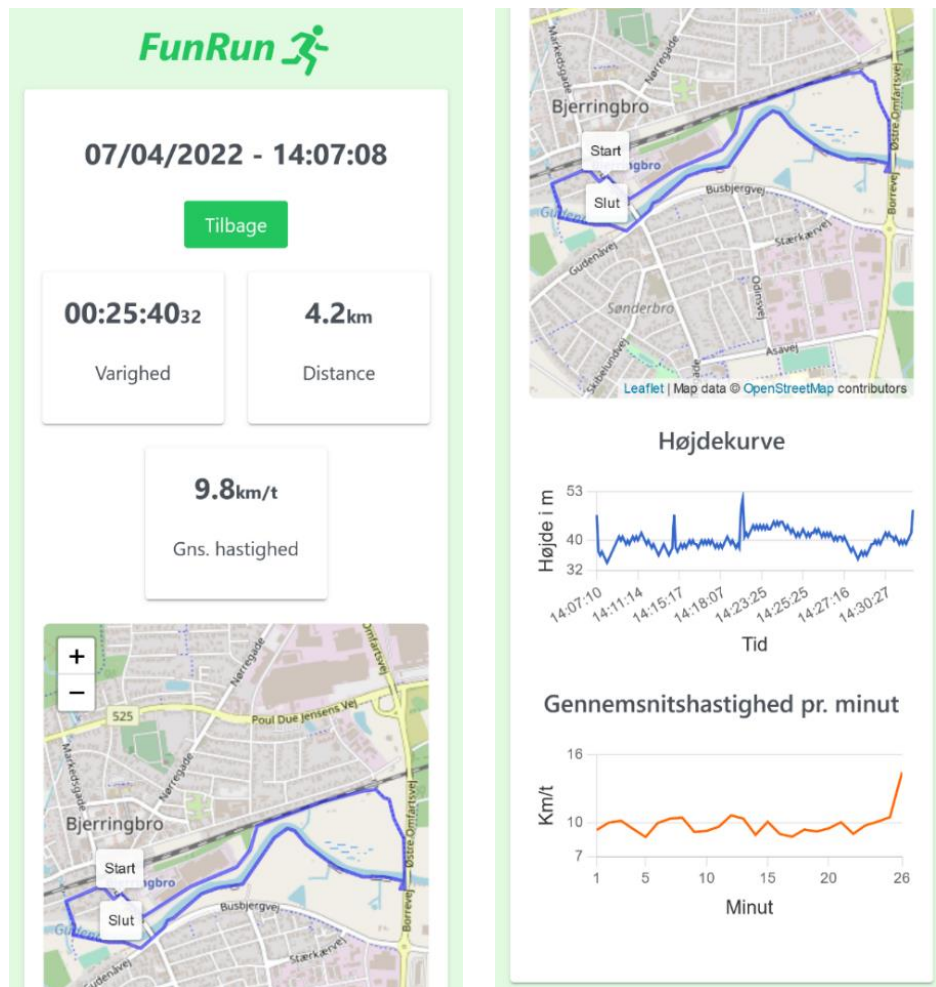
(Fuld kode: https://bit.ly/GraesholtFunRunPwaRun)
```

I denne kode forespørges informationen fra API'et, og når den ankommer bliver den gemt i objektet `refRun` som nogle af dom-elementerne er bundet til properties på. Sidens header bliver omregnet fra UTC tid fra databasen til lokal tid og dato, med Engelsk formatering. Efterfølgende bliver kortet sat op, og der loopes gennem turens `points` hovedsageligt for at gøre to ting. Først og fremmest extends `mapLatLngBounds` med hvert punkt for at lave de yderlighedspunkter brugeren er interesseret i at få vist på kortet. Derudover skabes en brugbar liste (`mapPointList`) over koordinater til fremvisning derpå. Efter loopet sættes kortets synsfelt, og derefter oprettes og konfigureres `runPolyline`, som visualiserer selve turen fra `mapPointList`, og `startTooltip` og `endTooltip`, som viser start- og slutpunkterne på kortet.

En mængde kode er udeladt her ved afmærkningen `[CODE OMITTED]`, for læselighed og forståelsens skyld. Koden herfra er omregning og formatering af de resterende databunde værdier på siden, disse værende `refRun.duration`, `refRun.distance`, `refRun.avgSpeedInMetersPerSecond`, `refAltitudePointList`, og `refAvgSpeedPerMinutePointList`. For de to sidste variabler loopes hen over data fra `refRun`, og dette formateres til visning i diagrammerne på siden.

## Resultat

Resultatet af denne udveksling af information er at en bruger kan se deres bevægelse på et kort, og bede deres enhed om at logge deres position til en central database. Efterfølgende er samme bruger i stand til at få præsenteret deres fulde bevægelse visuelt, samt information om varighed, hastighed, og mere, som vist nedenfor.



# Brugervejledninger

Velkommen til FunRun!

Denne brugervejledning er rettet mod nye brugere og brugere der ønsker et mere detaljeret indblik i brugen af systemet.

Punkterne i brugervejledningen er beregnede til at blive fulgt kronologisk.

## 1. Installation af app

FunRun kan installeres på en enhed via en browser, skønt nogle browsere ikke understøtter installation af apps på denne måde.

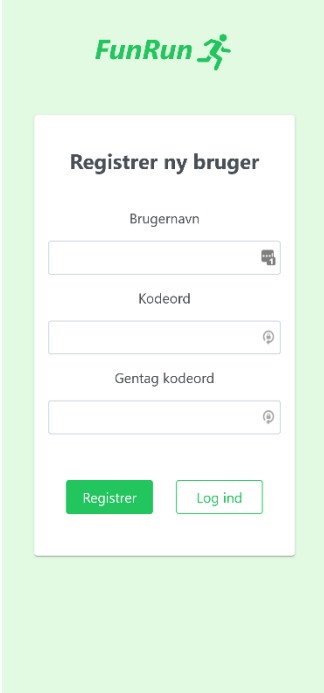
Denne guide går ud fra at brugeren bruger Google Chrome, som er gratis.

1. Naviger til adressen <https://bit.ly/GraesholtFunRun> på din enhed.  
Når siden er indlæst vil du se FunRuns "login"-side.
2. Tryk på de tre små prikker i øverste højre hjørne for at få en dropdown-menu med valgmuligheder.
3. Tryk på "Installer FunRun".
4. Bekræft at du gerne vil installere FunRun, i den pop-up dialog der vises.
5. Når appen er installeret, vil du kunne lokalisere den på din appoversigt som enhver anden app. Du er nu klar til at åbne FunRun, uafhængig af din browser.

## 2. Oprettelse af bruger

Dette punkt går ud fra at brugeren har installeret FunRun på deres enhed (se brugervejledning 1).

1. Åben appen.  
Når appen åbner vil du se en login skærm.
2. Tryk på "Registrer"-knappen nederst på siden.  
Du vil nu se FunRuns registreringsside.
3. Indtast et brugernavn og et kodeord.
4. Tryk på "Registrer"-knappen når dine oplysninger er tastet ind.  
Hvis brugernavnet allerede er i brug, vil du få dette at vide. I dette tilfælde vil du skulle vælge et andet brugernavn hvis du vil bruge systemet. Dit brugernavn skal mindst være 6 tegn.  
Hvis dine kodeord ikke er indtastet ens, vil du få dette at vide. Dit kodeord skal være mindst 8 tegn, og indeholde mindst ét stort, og ét småt bogstav, samt mindst ét tal eller specialtegn.



Registreringsside

Hvis oplysningerne bliver godtaget, vil du blive omdirigeret til login siden. Du er nu klar til at logge ind med din nye bruger.

### 3. Log ind

Dette punkt går ud fra at brugeren har registreret en FunRun bruger (se brugervejledning 2).

1. Åben appen.

Når appen åbner, vil du se en loginside.

2. Indtast de oplysninger du brugte da du registrerede dig som ny bruger.
3. Tryk på "Log ind"-knappen når dine oplysninger er tastet ind.  
Hvis oplysningerne bliver godtaget, vil du blive logget ind. Du er nu klar til at bruge systemet.

Når du vil logge din bruger af appen, er det så nemt som at trykke på den røde "Log ud"-knap.

### 4. Ny løbetur

Dette punkt går ud fra at brugeren har logget ind via en FunRun bruger (se brugervejledning 3).

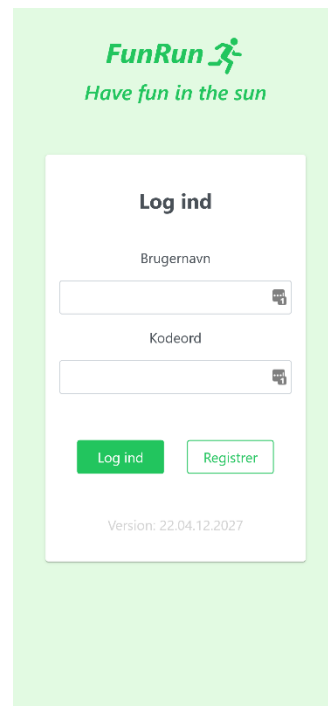
Når du ser oversigten over dine tidligere ture er du klar til at optage en ny løbetur.

1. Tryk på "Ny løbetur"-knappen øverst på siden.

Du vil nu blive omstillet til en side med et kort og en timer der står på nul.

Hvis det er første gang du ser denne side, vil du blive spurgt om FunRun må få adgang til din lokationsdata. Denne vil kun blive indsamlet i fremtiden når du kan se denne side, og det er kun dig der kan få dataene at se efterfølgende.

2. Når du er klar til at begynde, tryk på "Start løbetur"-knappen.  
Efter et kort øjeblik vil din enhed sende din position til FunRuns database og tegne din bevægelse på kortet løbende. Timeren vil også vise hvor længe du har løbet.  
Dette virker bedst hvis du slår WiFi fra på din enhed, og holder skærmen tændt og appen i fokus.



Loginside



Ny løbetur side



3. Når du er færdig med din tur, trykker du på "Stop løbetur"-knappen.

Du vil nu blive omdirigeret til en side hvor du kan se information om din netop afsluttede løbetur.

"Tilbage"-knappen vil returnere dig til oversigten over dine tidligere ture.

## 5. Mine løbeture

Dette punkt går ud fra at brugeren har optaget en løbetur (se brugervejledning 4).

1. Log ind i appen, eller tryk på "Tilbage"-knappen når du ser en færdig løbetur.

Du vil se en side med dine tidligere løbeture, markeret med dato og klokkeslæt.

Listen kan vise op til ti ture per side, og kalenderne over listen kan bruges til at afgrænse resultaterne via start og slutdato for viste ture.

Hvis du trykker på en løbetur i listen, vil du blive omdirigeret til en side med information om turen.

## 6. Vis løbetur

Dette punkt går ud fra at brugeren har optaget en løbetur (se brugervejledning 4).

1. Færdiggør en løbetur, eller tryk på en i listen over tidligere løbeture.

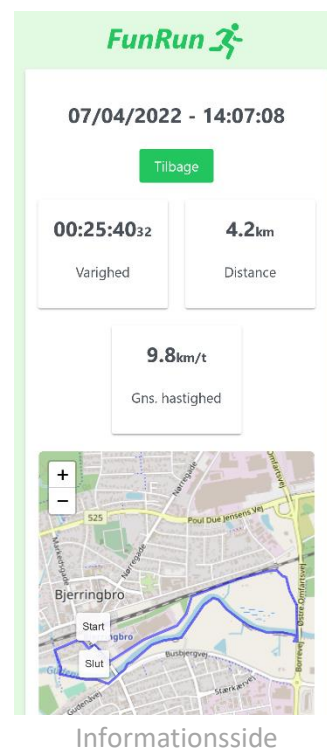
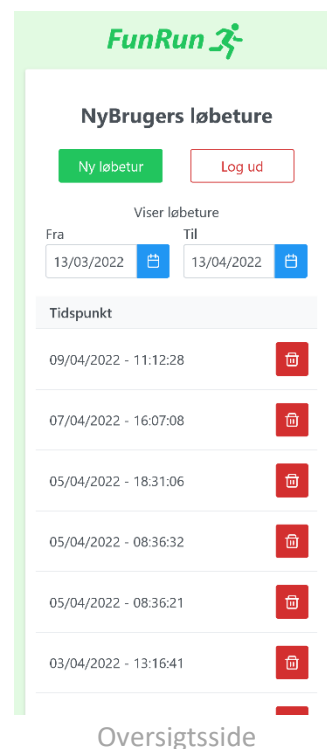
Du vil nu blive omdirigeret til en side hvor du kan se information om den givne løbetur.

Øverst på siden, under turens tidspunkt og sidens "tilbage"-knap, ses varighed, distance, og gennemsnitshastighed.

Under disse ses et kort hvor turen, samt dens start- og slutpunkter er markeret.

Nederst på siden ses to diagrammer. Det første viser en højdekurve over hvad din enhed har meddelt under turen, mens det andet viser gennemsnitshastigheden per minut turen har varet.

"Tilbage"-knappen vil returnere dig til oversigten over dine tidligere ture.



## 7. Slet løbetur

Dette punkt går ud fra at brugeren har kendskab til oversigtssiden (se brugervejledning 5).

1. Log ind i appen, eller tryk på "Tilbage"-knappen når du ser en færdig løbetur.  
Du vil se en side med dine tidligere løbeture, markeret med dato og klokkeslæt.  
Hver tur har i højre side en rød knap med en lille skraldespand.
2. Kontrollér at du ved hvilken tur du har med at gøre før du trykker på den røde knap.
3. Tryk på den røde knap.

Turen vil forsvinde fra listen, og nu være markeret som slettet i FunRuns database.

Slettet data er skjult for brugeren, men eksisterer stadig i FunRuns database. Hvis du ved et uheld har slettet en tur du gerne vil have gendannet, kontakt da kundeservice, og hvis muligt vil FunRuns personale forsøge at gendanne dine data.

FunRun tager ikke ansvar for slettet data der ikke umiddelbart kan fremfindes og gendannes, da dette er en tidskrævende proces.

# Kildeliste

What is an API?

<https://aws.amazon.com/what-is/api/>

(Sidst besøgt 12/04/2022)

What is Entity Framework?

<https://www.entityframeworktutorial.net/what-is-entityframework.aspx>

(Sidst besøgt 12/04/2022)

Introducing JSON

<https://www.json.org/json-en.html>

(Sidst besøgt 12/04/2022)

Hashing in Action: Understanding bcrypt

<https://auth0.com/blog/ hashing-in-action-understanding-bcrypt/>

(Sidst besøgt 12/04/2022)

SQL - Overview

<https://www.tutorialspoint.com/sql/sql-overview.htm>

(Sidst besøgt 12/04/2022)

The Declarative Nature of SQL

<https://vegibit.com/the-declarative-nature-of-sql/>

(Sidst besøgt 12/04/2022)

What are Progressive Web Apps?

<https://web.dev/what-are-pwas/>

(Sidst besøgt 12/04/2022)

Vue.js overview

<https://v1.vuejs.org/guide/overview.html>

(Sidst besøgt 12/04/2022)

What is Microsoft Azure: How Does It Work and Services

<https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure>

(Sidst besøgt 12/04/2022)

# Bilag 1

## Brugertestcases

FunRun systemet kan køre på to hovedplatforme, web og app. Hver test beskriver hvilke platforme den kan udføres på, og hver test bør udføres på alle egnede platforme før den er bestået.

Hvis en test fejler, ved at der sker andet end det forventede resultat, beskriv fejlen kortfattet i "Note/fejl"-kolonnen ud for det fejlede trin, og aflever papiret i udviklingsafdelingen. Kontroller venligst at navn på tester og software version (forefindes nederst på loginsiden) er udfyldt, og at platform er tydeligt markeret.

Til tests der kræver login, hvor andet ikke er specificeret, bruges brugertestoplysningerne:

Brugernavn: FunRunUserTesting

Kodeord: tQeWsEtRiTnYg5

### 1. Registrerings test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Tryk på "Registrer"-knappen.	System viser registreringssiden.	
4	Indtast et brugernavn der bruger den nuværende tid og følger konventionen: FRT-(år).(måned).(dag)-(time).(minut).(sekund).FRT, for eksempel "FRT-2022.04.12-17.24.05.FRT"*	Ingen ændring.	
5	Indtast kodeord: tQeWsEtRiTnYg5 i begge felter.	Ingen ændring.	
6	Tryk på "Registrer"-knappen.	System viser success-besked efterfulgt af loginsiden.	

\*Testbrugere oprettet per denne konvention bliver slettet automatisk i databasen.

## 2. Brugernavn i brug test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Tryk på "Registrer"-knappen.	System viser registreringssiden.	
4	Indtast brugernavn: FunRunUserTesting	Ingen ændring.	
5	Indtast kodeord: tQeWsEtRiTnYg5 i begge felter.	Ingen ændring.	
6	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	

## 3. Brugernavn validering test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Tryk på "Registrer"-knappen.	System viser registreringssiden.	
4	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	
5	Indtast et brugernavn med mindre end 6 karakterer.	Ingen ændring.	
6	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	
7	Indtast et brugernavn med mere end 30 karakterer.	Ingen ændring.	

8	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	
9	Indtast et brugernavn mellem 8 og 30 karakterer der indeholder "#".	Ingen ændring.	
10	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	
11	Indtast brugernavn: FunRunUserTesting	Ingen ændring.	
12	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	

## 4. Kodeord validering test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Tryk på "Registrer"-knappen.	System viser registreringssiden.	
4	Indtast brugernavn: FunRunUserTesting	Ingen ændring.	
5	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
6	Indtast et kodeord med mindre end 8 karakterer.	Ingen ændring.	
7	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
8	Indtast et kodeord med mere end 99 karakterer.	Ingen ændring.	
9	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
10	Indtast et kodeord med mellem 8 og 99 karakterer der kun indeholder store bogstaver.	Ingen ændring.	
11	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
12	Indtast et kodeord med mellem 8 og 99 karakterer der	Ingen ændring.	

	kun indeholder små bogstaver.		
13	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
14	Indtast et kodeord med mellem 8 og 99 karakterer der indeholder store og små bogstaver, men hverken tal eller specialkarakterer.	Ingen ændring.	
15	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
16	Indtast et kodeord med mellem 8 og 99 karakterer der indeholder store og små bogstaver, mindst et tal, og "#".	Ingen ændring.	
17	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
18	Indtast kodeord: tQeWsEtRiTnYg5.-_@!? i kun øverste felt.	Ingen ændring.	
19	Tryk på "Registrer"-knappen.	Siden viser en kodeord-fejl.	
20	Indtast kodeord: tQeWsEtRiTnYg5.-_@!? i nederste felt.	Ingen ændring.	
21	Tryk på "Registrer"-knappen.	Siden viser en brugernavn-fejl.	

## 5. Loginside test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Indtast brugernavn: FunRunUserTesting	Ingen ændring.	
4	Indtast et vilkårligt, men ukorrekt kodeord.	Ingen ændring.	

5	Tryk på "Log ind"-knappen.	Siden viser en fejl.	
6	Indtast kodeord: tQeWsEtRiTnYg5	Ingen ændring.	
7	Tryk på "Log ind"-knappen.	System viser oversigtssiden.	

## 6. Omdirigering test

Platform: web

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Start FunRun.	System starter på enten login- eller oversigtssiden.	
2	(Hvis på oversigtssiden) Tryk på "Log ud"-knappen.	System viser loginsiden.	
3	Tryk på "Registrer"-knappen.	System viser registreringssiden.	
4	Genindlæs siden.	System omdirigerer til loginsiden.	
5	Log ind på FunRun.	System viser oversigtssiden.	
6	Naviger til loginsiden, uden at logge ud (slet "runs" i adressen).	System omdirigerer til oversigtssiden.	

## 7. Ny løbetur test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	
2	Tryk på "Ny løbetur"-knappen	System viser "Ny løbetur"-siden. Kort opdaterer position periodisk.	
3	Tryk på "Start løbetur"-knappen.	Knap bliver grå og ændrer tekst.	



4	Vent på at løbetur starter.	Knap bliver rød og ændrer tekst. Timer starter.	
5	Gå en kort tur i mindst 30 sekunder.	Kort opdaterer position periodisk, og tur indtegnes derpå.	
6	Tryk på "Stop løbetur"-knappen.	System viser informationssiden.	

## 8. Oversigtsside test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	
2	(Hvis ingen løbeture i oversigten) Lav en løbetur så mindst en vises.	Mindst en løbetur vises i oversigten.	
3	Sæt begge kalenderdatoer så de viser et tidsrum før seneste løbetur fandt sted.	Seneste løbetur forsvinder fra oversigten.	
4	Sæt begge kalenderdatoer så seneste løbetur vises.	Mindst en løbetur vises i oversigten.	
5	Sæt begge kalenderdatoer så de viser et tidsrum efter seneste løbetur fandt sted.	Seneste løbetur forsvinder fra oversigten.	
6	Sæt begge kalenderdatoer så seneste løbetur vises.	Mindst en løbetur vises i oversigten.	
7	Sæt begge kalenderdatoer til den dato seneste løbetur fandt sted.	Seneste løbetur vises stadig i oversigten.	

## 9. Informationsside test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	

2	(Hvis ingen løbeture i oversigten) Lav en løbetur så mindst en vises.	Mindst en løbetur vises i oversigten.	
3	Tryk på en løbetur i oversigten.	System viser informationssiden.	
4	Kontrollér at dato i overskrift stemmer overens med dato på oversigtssiden.	Ingen ændring.	
5	Kontrollér at de tre felter øverst på siden viser data.	Ingen ændring.	
6	Kontrollér at kortet viser løbeturen, samt start- og slutpunkter.	Ingen ændring.	
7	Kontrollér at kortet starter centreret på løbeturen, og at hele løbeturen kan ses når siden hentes.	Ingen ændring.	
8	Kontrollér at begge diagrammer under kortet viser data fra turen.	Ingen ændring.	

## 10. Ingen punktdata test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	
2	Tryk på "Ny løbetur"-knappen	System viser "Ny løbetur"-siden. Kort opdaterer position periodisk.	
3	Tryk på "Start løbetur"-knappen.	Knap bliver grå og ændrer tekst.	
4	Naviger tilbage via systemnavigation før knappen bliver rød og ændrer tekst.	System viser oversigtssiden.	
5	Tryk på seneste løbetur i oversigten.	System viser informationssiden. Overskriften er "Ingen punktdata". (Hvis overskriften viser en dato, har turen nået at generere et eller flere punkter. Gentag trin 2 til 5.)	

## 11. Slet løbetur test

Platform: web, app

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	
2	(Hvis ingen løbeture i oversigten) Lav en løbetur så mindst en vises.	Mindst en løbetur vises i oversigten.	
3	Tryk på en løbeturs røde "Slet"-knap.	Listen genhentes uden løbeturen.	

## 12. Informationstyveri test

Platform: web

Tester:

Version:

Trin	Instruktion	Forventet resultat	Note/fejl
1	Log ind på FunRun.	System viser oversigtssiden.	
2	(Hvis ingen løbeture i oversigten) Lav en løbetur så mindst en vises.	Mindst en løbetur vises i oversigten.	
3	Tryk på en løbetur i oversigten.	System viser informationssiden.	
4	Notér tallet til sidst i sidens adresse i noten til dette trin.	Ingen ændring.	
5	Tryk på "Tilbage"-knappen.	System viser oversigtssiden.	
6	Tryk på "Log ud"-knappen.	System viser loginsiden.	
7	Indtast brugernavn: FunRunTestingUnauthorized	Ingen ændring.	
8	Indtast kodeord: tQeWsEtRiTnYg5	Ingen ændring.	

9	Tryk på "Log ind"-knappen.	System viser oversigtssiden.	
10	Naviger til informationssiden for den anden brugers løbetur (udskift "runs" i adressen med "run/[tallet du noterede i punkt 4]").	System viser informationssiden. Overskriften er "Ugyldigt runId!" efter indlæsning.	