

Sport To Go

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | : | 26-06=2021 |  |  |
| Auteur | : | Niels Riezebos |  |  |

Revisiehistorie

Geef per nieuwe versie aan wat er in de versie is toegevoegd/veranderd/verwijderd.

Inhoudsopgave

[1 Inleiding 3](#_Toc75612756)

[2 Overzicht 3](#_Toc75612757)

[3 Use cases 4](#_Toc75612758)

[3.1 Actoren 5](#_Toc75612759)

[3.2 Use case templates 5](#_Toc75612760)

[3.3 Wireframes 9](#_Toc75612761)

[4 Domeinmodel 10](#_Toc75612762)

[5 Technologieën 11](#_Toc75612763)

[6 Overdracht 12](#_Toc75612764)

[7 Referenties 13](#_Toc75612765)

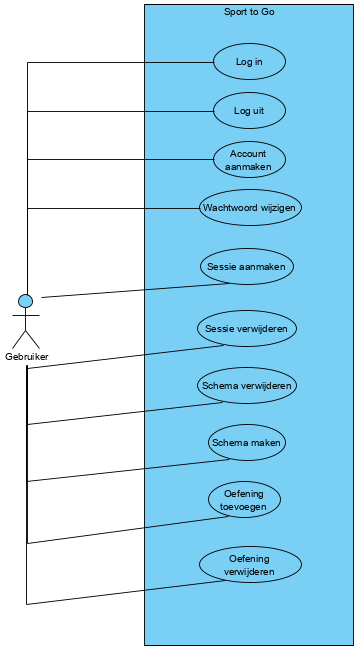
# Inleiding

Door de corona pandemie zijn er veel beperkingen op het dagelijks leven gelegd. Afspraken maken om te kunnen winkelen, maskers dragen als je erop uit gaat. Sommige bedrijven hadden hun deur compleet moeten sluiten. Sportscholen vallen onder die bedrijven. Mensen die normaal dagelijks naar de sportschool gingen moeten andere manieren vinden om zichzelf in vorm te houden. Een groot onderdeel van sporten is de consistentie waarmee je sport. Een dat is lastig vol te houden als je niet naar de sportschool kunt gaan. Deze website zal je de optie geven om zelf een sportschema in elkaar te zetten met oefeningen die je allemaal thuis kunt doen met wat halters en vrije tijd.

# Overzicht

Met dit project wil ik een duidelijke en gebruikersvriendelijke website maken waarmee mensen hun eigen sportschema op kunnen stellen. Ik wil vooral focussen op een groot aantal oefeningen waar mensen uit kunnen kiezen. Het doel is om gemakkelijk sport schema's aan te maken met de oefeningen die zijn aangeboden door de website. Deze schema's moeten in te plannen zijn door middel van sessies. Deze sessies zijn terug te vinden op het home scherm van de website. Ook wil ik persistentie goed in het systeem hebben zitten zodat alles wat je aanpast aan je schema's en sessies word opgeslagen.

# Use cases



Het eindresultaat zal een website worden waarin mensen op een eigen account verschillende schema’s kunnen maken. De belangrijkste deelresultaten van dit project zijn de volgende use cases:

In- en uitloggen. De gebruiker heeft de mogelijkheid om in en uit te loggen.

Account aanmaken. De gebruiker kan een eigen account aanmaken.

Wachtwoord wijzigen. De gebruiker is in staat zijn/haar wachtwoord te wijzigen.

Sessie aanmaken. De gebruiker heeft de mogelijkheid om een sessie aan te maken door middel van een schema en zelfgekozen data.

Sessie verwijderen. De gebruiker is in staat om eigen sessies te verwijderen.

Schema’s toevoegen. De gebruiker is in staat nieuwe schema’s aan te maken.  
Schema’s verwijderen. De gebruiker is in staat om schema’s te verwijderen.

Oefeningen toevoegen. De gebruiker is in staat om oefeningen aan bestaande schema’s toe te voegen

Oefeningen verwijderen. De gebruiker kan oefeningen van bestaande schema’s verwijderen

## Actoren

Naam: Gebruiker

Beschrijving: De gebruikers zijn de mensen die gebruik maken van de website

## Use case templates

|  |  |
| --- | --- |
| Id | UC1 |
| Naam | Log in |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om in te loggen met zijn/haar inloggegevens |
| Preconditie | - |
| Scenario | 1. Systeem vraagt om inloggegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1 4. Als de ingevulde gegevens niet kloppen 4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Systeem gaat naar de thuis pagina |
| Postconditie | De actor is ingelogd op zijn/haar account |

De use case voor het inloggen.

|  |  |
| --- | --- |
| Id | UC2 |
| Naam | Log uit |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om uit te loggen |
| Preconditie | - |
| Scenario | 1. Actor drukt op de log uit knop  2. Systeem logt de gebruiker uit  3. Systeem gaat terug naar de login pagina |
| Postconditie | De actor is uitgelogd van zijn/haar account |

De use case voor het uitloggen.

|  |  |
| --- | --- |
| Id | UC3 |
| Naam | Account aanmaken |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een nieuw account aan te maken door middel van een (i)gebruikersnaam, (i)emailadres en (i)wachtwoord |
| Preconditie | Het emailadres staat niet al geregistreerd |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Als het gegeven email adres al bestaat  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Het account is aangemaakt  6. Het systeem geeft een bericht dat het account is aangemaakt |
| Postconditie | De actor heeft een nieuw account aangemaakt |

De ues case voor een account aanmaken.

|  |  |
| --- | --- |
| Id | UC4 |
| Naam | Wachtwoord wijzigen |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om zijn/haar wachtwoord te veranderen |
| Preconditie | Het nieuwe wachtwoord moet anders zijn dan het vorige wachtwoord Het gegeven oude wachtwoord moet overeenkomen met het huidige wachtwoord |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Als het huidige wachtwoord niet overeen komt met het gegeven oude wachtwoord  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Als de opgegeven nieuwe wachtwoorden niet overeen komen  5.1 Systeem geeft foutmelding  5.1 Systeem gaat terug naar stap 1  6. Het systeem wijzigd het wachtwoord  7. Het systeem geeft een bericht dat het wachtwoord is gewijzigd |
| Postconditie | Het wachtwoord van de actor is nu veranderd |

De use case voor een wachtwoord wijzigen.

|  |  |
| --- | --- |
| Id | UC5 |
| Naam | Sessie aanmaken |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een sessie aan te maken d.m.v schema’s |
| Preconditie | Er moet een schema aanwezig zijn |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Als de gegeven datum al verlopen is  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Als de gegeven eindtijd voor de begintijd ligt  5.1 Systeem geeft foutmelding  5.1 Systeem gaat terug naar stap 1  6. Als de gegeven sessie al bestaat  6.1 Systeem geeft foutmelding  6.2 Systeem gaat terug naar stap 1  7. Het systeem maakt de sessie aan  8. Het systeem print de sessie op het thuisscherm |
| Postconditie | De actor heeft een sessie aangemaakt |

De use case om een sessie aan te maken.

|  |  |
| --- | --- |
| Id | UC6 |
| Naam | Sessie verwijderen |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een sessie te verwijderen |
| Preconditie | Er moet een sessie aanwezig zijn |
| Scenario | 1. Systeem vraagt om gegevens  2. Als er geen sessies aanwezig zijn  2.1 Systeem geeft geen mogelijkheden om iets te verwijderen  2.2 systeem gaat terug naar stap 1  3. Actor vult gegevens in  4. Als niet alle gegevens zijn ingevuld  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Systeem verwijderd de gegeven sessie  6. Systeem geeft melding dat de sessie is verwjiderd |
| Postconditie | De actor heeft een sessie verwijderd |

De use case om een sessie te verwijderen.

|  |  |
| --- | --- |
| Id | UC7 |
| Naam | Schema maken |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een schema op te stellen uit een of meerdere oefeningen |
| Preconditie | - |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Als het schema al bestaat  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Systeem maakt schema aan |
| Postconditie | De actor heeft een schema aangemaakt |

Use case om een schema aan te maken.

|  |  |
| --- | --- |
| Id | UC8 |
| Naam | Schema verwijderen |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een gekozen schema te verwijderen |
| Preconditie | - |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Systeem verwijderd gegeven schema |
| Postconditie | De actor heeft een schema verwijderd |

Use case om een schema te verwijderen.

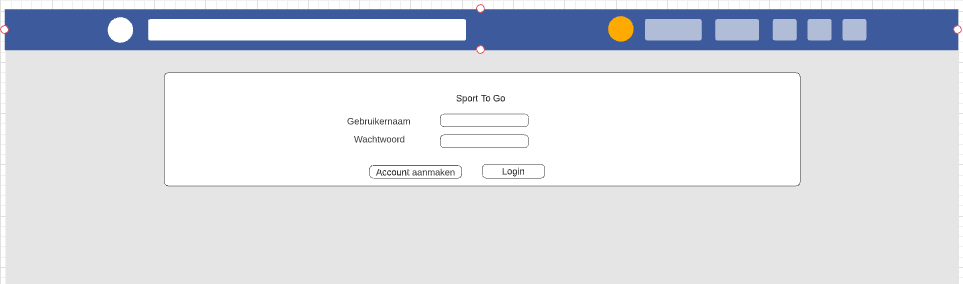
|  |  |
| --- | --- |
| Id | UC9 |
| Naam | Oefening toevoegen |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een oefening aan een schema toe te voegen door middel van een gekozen schema, gewicht en sethoeveelheid |
| Preconditie | Er moet een schema aanwezig zijn om oefeningen in te stoppen |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult gegevens in  3. Als niet alle gegevens zijn ingevuld  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Als de oefening al in dit schema staat  4.1 Systeem geeft foutmelding  4.2 Systeem gaat terug naar stap 1  5. Systeem voegt de gekozen oefening toe aan het schema |
| Postconditie | De actor heeft een oefening aan het gekozen schema toegevoegd |

Use case om een oefening toe te voegen.

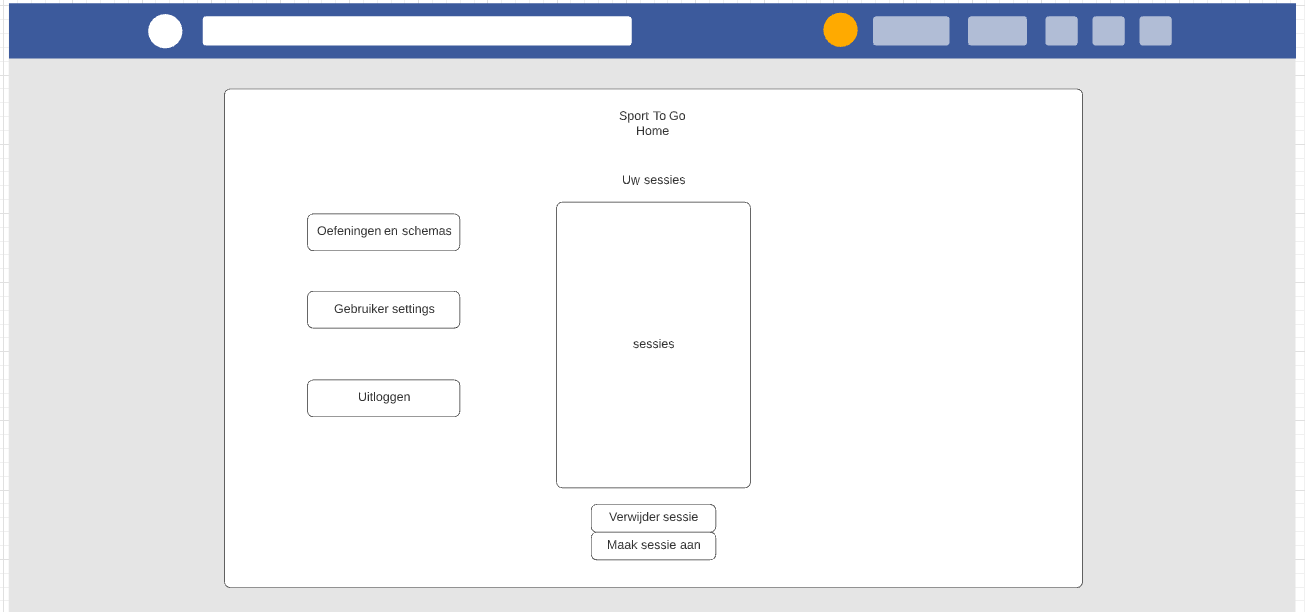
|  |  |
| --- | --- |
| Id | UC10 |
| Naam | Oefening verwijderen |
| Actoren | Gebruiker |
| Beschrijving | De actor is in staat om een oefening van een schema te verwijderen |
| Preconditie | Er moet een oefening aanwezig zijn |
| Scenario | 1. Systeem vraagt om gegevens  2. Actor vult schema in  3. Als er geen oefeningen in het gegeven schema zitten  3.1 Systeem geeft foutmelding  3.2 Systeem gaat terug naar stap 1  4. Actor vult oefening in  5. Systeem print oefening op scherm  6. Systeem verwijderd de oefening van het schema |
| Postconditie | De actor heeft een oefening verwijderd |

Use case om een oefening te verwijderen.

## Wireframes



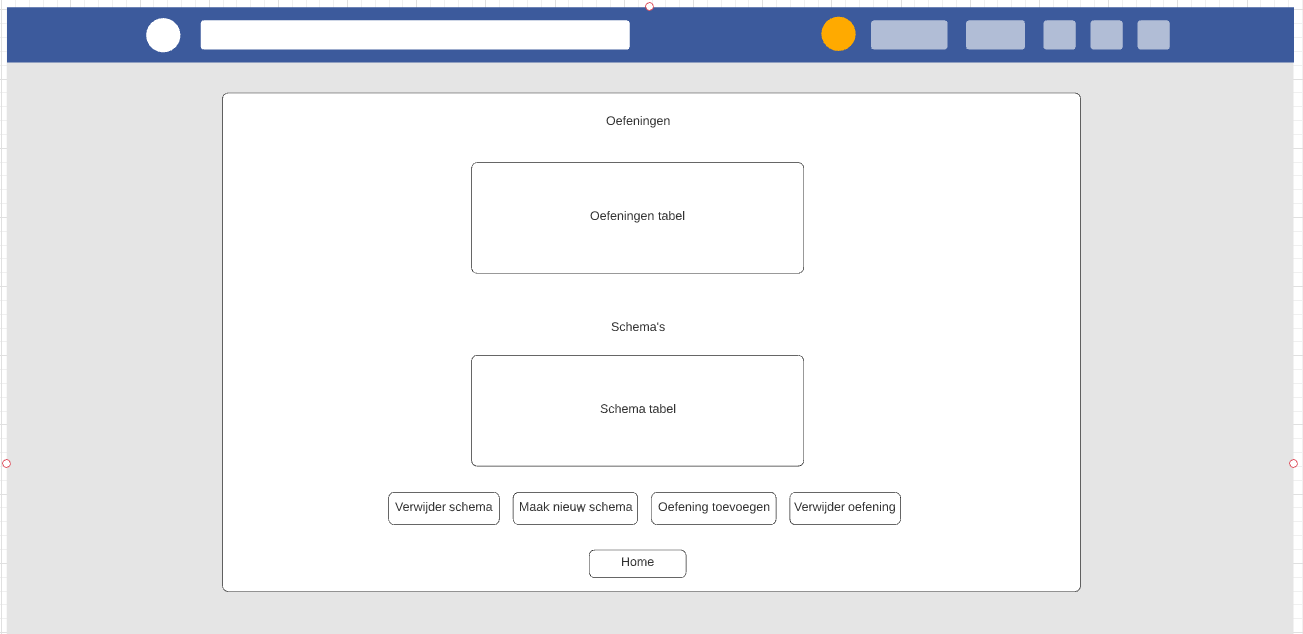
Login scherm.



Home scherm.

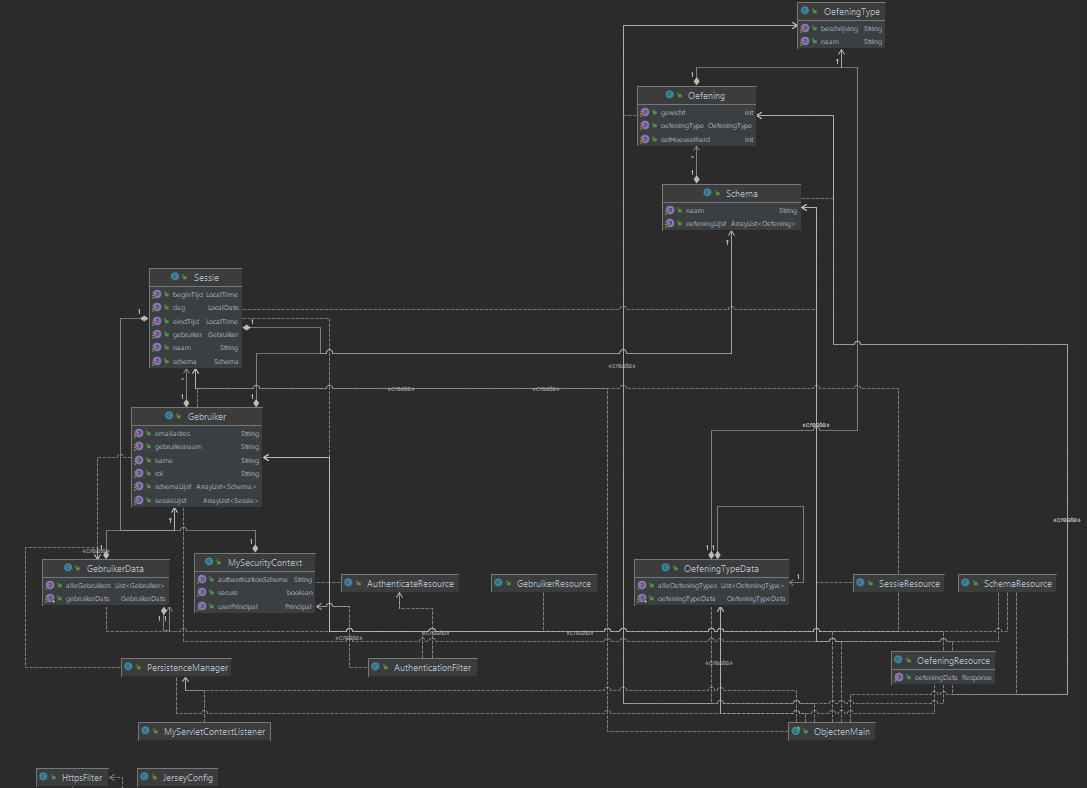


Gebruiker settings scherm.



Oefeningen en schema’s scherm.

# Domeinmodel



Domeinmodel met alle attributen.

(Zie SportToGo/documentatie voor meer diagrammen)

|  |  |
| --- | --- |
| Entiteit | Beschrijving |
| Gebruiker | Domeinklas voor de gebruiker |
| Oefening | Domeinklas voor een oefening instantie |
| OefeningType | Domeinklas voor alle oefeningen typen die in het systeem staan |
| Schema | Domeinklas voor alle schema’s |
| Sessie | Domeinklas voor alle sessies |
| MyServletContextListener | Klasse voor alle servlet listeners |
| GebruikerData | Klasse om alle huidige gebruikers te bewaren |
| OefeningTypeData | Klasse om alle huidige oefeningtypes te bewaren |
| PersistenceManager | Klasse om huidige gebruikers en huidige oefeningtypes op te slaan in een Azure blobstorage |
| AuthenticationFilter | Een filter om de JWT van elke request te checken |
| HttpsFilter | Een filter om de http verbinding om te zetten in een https verbinding |
| MySecurityContext | Klasse om de huidige gebruiker van de token te authenticeren en terug te vinden |
| JerseyConfig | Een klasse die voor navigatie van de URI requests zorgt |
| ObjectenMain | Een main die dummydata aanmaakt |
| AuthenticateResource | De resource om de gebruiker te authenticeren en om een token aan te maken |
| GebruikerResource | De resource die alle gebruiker gerelateerde requests afhandeld |
| OefeningResource | De resource die alle oefening gerelateerde reqeusts afhandeld |
| SchemaResource | De resource die alle schema gerelateerde reqeusts afhandeld |
| SessieResource | De resource die alle sessie gerelateerde reqeusts afhandeld |

Business rules:

Er mag maar een gebruiker per email geregistreerd staan.

Er mogen geen twee dezelfde oefeningen in een schema zitten.

De naam van een sessie moet uniek zijn.

De naam van een schema moet uniek zijn.

# Technologieën

Tomcat.

Om dit project online te krijgen had ik een manier nodig om met servlets te praten. In de eerste paar lessen van Back-end leerde we om deze servlets zelf te maken, maar het was erg snel duidelijk dat deze manier niet echt praktisch zou zijn voor IPASS. Gelukkig kwam school aanzetten met Tomcat. Dit is een servlet container of web container. Via Tomcat is het mogelijk om met servlets te interacteren. Er waren ook andere manieren om zelf een server op te zetten, maar voor iemand die nog nooit met servers en servlets heeft gewerkt was het makkelijker om de handleidingen van school te volgen.

Tomcat servlets hebben iets genaamd een 'lifecycle'. Deze begint wanneer de servlet klasse word geladen en eindigd wanneer de servlet word afgesloten. Alles wat plaatsvind hier tussen valt onder deze 'lifecycle', hieronder vallen de requests die een gebruiker opstuurt tijdens het gebruiker van de server.

Met methodes genaamd 'listeners' is het mogelijk om de server aan te spreken wanneer specifieke dingen gebeuren. Zoals wanneer de server start print je "Server gestart" in de console.

Een belangrijk deel van Tomcat is dat het niet online is. De server runt lokaal op je laptop of pc, dus het is geen manier om je project online te zetten. Wat het wel is, is makkelijk om op te zetten voor je project en samen met de handleidingen van school was het de logische keuze om te maken voor een beginner programmeur zoals ik.

Versie: Tomcat 9.0.45

Heroku.

Heroku is een 'cloud platform' wat de mogelijkheid bied om apps en websites online te zetten. Waar Tomcat lokaal een server biedt om aan je project te werken zet Heroku het online. Zelf niet voor de server hoeven zorgen waar je project om draait is erg handig voor een eerstejaars student. Heroku gebruikt Git om applicaties online te zetten, vandaar dat je je github repository moet linken voordat je je project online kunt zetten. Net als met Tomcat koos ik voor Heroku omdat school al handleidingen klaar had staan om er gebruik van te maken. Aangezien ik zelf geen manier had om mijn project online te zetten was Heroku de meest logische keuze.

Versie: 9.0.39.0

Azure blobstorage.

De Azure blobstorage is een manier om data op te slaan voor je project. Azure bied een hele hoop functies aan waaronder persistentie. Om deze persistentie te bereiken heb je een paar dingen nodig. Als eerst moet je een azure account hebben, daarnaast zal je een opslagaccount aan moeten maken. Nadat je een opslagaccout hebt aangemaakt kun je een blob storage aanmaken. Deze blob storage kan blobs bewaren. Blobs zijn een soort containers waarin je objecten kunt bewaren. Verbinden met een blobstorage doe je door middel van een SASTOKEN. In je project staat een methode die verbinding maakt met de blobcontainer die je gebruikt om data op te slaan. Wanneer je iets in de blobcontainer opslaat word alle data erin overschreven. Het is dus makkelijk om data te persisteren. De reden dat ik geen SQL database opzette of alles in text bestanden opsloeg is net als bij Heroku en Tomcat: school had meerdere voorbeelden en handleidingen klaarstaan. Daarnaast wist ik dat data in text bestanden opslaan erg snel tot errors leidt.

Version 12.6.0

# Overdracht

Welkom in de readme van SportToGo.

In deze readme zal ik uitleggen hoe je dit project werkende kan krijgen op jouw eigen pc/laptop, waar de integratie tests te vinden zijn en hoe je gebruik maakt van de website.

Om te beginnen moet je Tomcat instellen in IntelliJ. Start IntelliJ op nadat je Tomcat hebt geinstalleerd en ergens apart hebt staan. In het start-up scherm druk op 'Customize' -> ' All settings' ->  'Build, Execution, Deployment' -> Application Servers. Druk hier op de + en voeg Tomcat Server toe (als het goed is weet je zelf waar je Tomcat op je pc hebt staan). Druk dan op OK.

Nadat je Tomcat hebt geconfigureerd moet je het ook nog toevoegen aan de run configuration van het project. Dit doe je door op Run/Debug Configurations te drukken. Je krijgt dan een window waarin je jouw run configurations hebt. Druk op + in de linker bovenste hoek. Zoek Tomcat Server in de lijst en kies de local variant. Nadat je de Tomcat server hebt toegevoegd ga naar de Deployment tab en druk op + onder Deploy at server startup.  Selecteer hier war exploded en zorg ervoor dat de Application context op / staat.'

Ga dan terug naar de Server tab en stel daar de volgende instellingen in: On Update action: Redeploy, On frame deactivation: Update resources.  
Het project is nu klaar om gestart te worden.

Om bij de integratie tests te komen ga naar: http://localhost:8080/integratieLogin.html

Op elke pagina staan beschrijvingen van de business rules waar de tests aan moeten voldoen. De POST/DELETE/PUT methodes die worden aangeroepen zijn hetzelfde als de methodes die in het project worden gebruikt.

# Referenties

*How Heroku Works | Heroku Dev Center*. (z.d.). Heroku. Geraadpleegd op 26 juni 2021, van <https://devcenter.heroku.com/articles/how-heroku-works>

*An introduction to Tomcat servlet interactions*. (z.d.). MuleSoft. Geraadpleegd op 26 juni 2021, van <https://www.mulesoft.com/tcat/tomcat-servlet>