Sprint – report

Start date	30/11/2025
End date	1/12/2026
Sprint #	Sprint 2/3
Group	8 Sneaky Golem
Students	Xander UijtdeHaag, Bart Geijtenbeek, Boaz Vaneveld, Rick de Cuijper, Valentijn van Grunsven
Topic	Organization and Motivation for Maintenance of Sustainable Schoolyards Organisatie en Motivatie voor Onderhoud van Duurzame Schoolpleinen

Inhoud

Related System requirements (sprint backlog)

CALENDAR (Agenda & Taken)

Front-end

To Do

Schatkaart-pagina synchroniseren met kalender

- Schatkaart-pagina bijwerken zodat deze de actuele kalenderdata gebruikt
- *Assignee:* Rick

Kalender code refactoren

- Kalendercode opschonen
- Componenten opsplitsen voor betere structuur
- Code duplicatie verminderen
- *Assignee:* Rick

In Progress

Front-end ↔ Back-end integratie

- Koppeling maken tussen front-end en back-end (API connecties)
- *Assignees:* Rick, Boaz

In Review

- Er staan op dit moment geen taken in review

Done

Kalender – Functionaliteit

- Weken- en maandenweergave geïmplementeerd in de kalender
  Assignee: Rick
- Kalender met taken gebouwd
  Assignee: Rick
- Taken aanmaken, bewerken en verwijderen in de kalender
  Assignee: Rick
- Drag & drop-systeem voor taken in de kalender
  Assignee: Rick

Taskpool

- Taskpool gemaakt voor taken buiten de kalender
  Assignee: Rick
- Taken aanmaken, bewerken en verwijderen in de taskpool
  Assignee: Rick

Styling & UX

- Kalenderpagina gestyled volgens het Figma-prototype
  Assignee: Rick
- Consistente UI-componenten toegepast binnen de kalender
  Assignee: Rick
- Basis foutafhandeling en gebruikersfeedback toegevoegd
  Assignee: Rick

Back-end

To Do

Schatkaart-pagina synchroniseren met kalender

- API maken zodat kalenderdata doorgegeven kan worden aan de schatkaartpagina
- Taakdata ophalen en formatteren voor frontend

- Assignee: Boaz

Kalendercode refactoren (Backend)
- Endpoints opschonen en logica herstructureren
- Data-validatie centraliseren
- Eventuele duplicatie in databasequeries verminderen
- Assignee: Boaz

Schatkaart-pagina backend
- API maakt kalenderdata beschikbaar voor de schatkaartpagina
- Controle van juiste mapping tussen taken en schatkaart-elementen
- Assignee: Boaz

In Progress

Front-end ↔ Back-end integratie
- API-koppelingen implementeren zodat frontend taken kan ophalen, toevoegen, bewerken en verwijderen
- API voor CRUD taken in de kalender
- Data beschikbaar stellen voor frontend
- Endpoint testen met Postman
- Assignees: Rick, Boaz

Taskpool
- API voor taskpool CRUD-functionaliteit
- Taskpool data opslaan en ophalen via backend
- Assignee: Boaz

Done

Kalender – Functionaliteit
- Taken correct opslaan in database
- Assignee: Boaz

AVATAR CUSTOMIZATION

Front-end

Avatar Customization

To Do
- Avatar onderdelen toevoegen in UI (haar, kleding, accessoires)
- Dropdowns / sliders voor avatar eigenschappen
- Assignee: Valentijn

In Progress
- Real-time preview van avatar wijzigingen implementeren
- Avatar kleuren en opties dynamisch laden uit API
- Assignee: Valentijn

In Review
- UI getest met meerdere browsers / schermgroottes

- Assignee: Valentijn

Done
- Avatar customisatiepagina opgezet volgens Figma
- Selectie van avatar onderdelen werkt interactief
- Assignee: Valentijn

Back-end

Avatar Customization
To Do
- API endpoint voor avatar ophalen per gebruiker
- Database-model voor avatar instellingen maken
- Validatie van avatar data (geldige opties)
- Assignee: Xander

In Progress
- N/A

Done
- Avatar wijzigingen correct opslaan in database
- API retourneert avatar data correct naar frontend
- API endpoint voor avatar opslaan / bijwerken
- Assignee: Xander

Login

Front-end

Login
To Do
- Login-formulier component maken
- Validatie van gebruikersnaam en wachtwoord in frontend
- Assignee: Bart

In Progress
- Feedback tonen bij verkeerde inloggegevens
- JWT token/ cookie opslaan
- Assignee: Bart

Done
- Login-formulier werkend en gestyled volgens Figma
- Wachtwoord zichtbaar/verborgen toggle geïmplementeerd
- Assignee: Bart

Back-end

Login
To Do
- Backend login endpoint (ExpressJS)
- Hashing van wachtwoorden implementeren (bcrypt)
- Assignee: Bart

In Progress
- Cookie aanmaken en terug sturen
- Assignee: Bart

Done

- Gebruiker kan succesvol inloggen via backend
- JWT-token/ cookie correct aangemaakt en teruggestuurd
- Authenticatie token (JWT)/ cookie genereren bij succesvolle login
- Middleware voor beveiligde routes implementeren
- Assignee: Bart

1. Kalender & Schatkaart – Requirements

1.1 Functionele Requirements (Front-end)

- De kalender moet weken- en maandenweergave ondersteunen.
- Gebruikers moeten taken kunnen aanmaken, bewerken en verwijderen via de kalender.
- Taken moeten via drag & drop verplaatst kunnen worden binnen de kalender.
- Er moet een taskpool bestaan voor taken die (nog) niet aan een datum gekoppeld zijn.
- De schatkaart-pagina moet automatisch synchroniseren met actuele kalenderdata.
- De kalenderpagina moet gestyled zijn volgens het Figma-prototype.
- De UI moet consistente componenten gebruiken.
- Het systeem moet basis foutafhandeling en gebruikersfeedback tonen.

1.2 Niet-functionele Requirements (Front-end)

- De kalendercode moet opgeschoond en modulair opgebouwd zijn.
- Code duplicatie moet zoveel mogelijk vermeden worden.
- Componenten moeten logisch gesplitst zijn voor onderhoudbaarheid.

1.3 Functionele Requirements (Back-end)

- De backend moet CRUD-functionaliteit bieden voor kalendertaken.
- Taken moeten correct worden opgeslagen in de database.
- De backend moet een API aanbieden die kalenderdata beschikbaar stelt voor de schatkaart-pagina.
- Taakdata moet correct geformatteerd worden voor frontend-gebruik.
- De backend moet CRUD-functionaliteit ondersteunen voor de taskpool.
- API-endpoints moeten getest worden (bijv. met Postman).

1.4 Niet-functionele Requirements (Back-end)

- Endpoints moeten opgeschoond en logisch herstructureerd worden.
- Data-validatie moet gecentraliseerd plaatsvinden.
- Duplicatie in databasequeries moet geminimaliseerd worden.

2. Front-end ↔ Back-end Integratie – Requirements

- De frontend moet via API’s taken kunnen ophalen, toevoegen, bewerken en verwijderen.
- De integratie moet real-time consistente data tonen tussen frontend en backend.
- Foutmeldingen vanuit de backend moeten correct worden afgehandeld in de frontend.

3. Avatar Customization – Requirements

3.1 Functionele Requirements (Front-end)

- Gebruikers moeten hun avatar kunnen aanpassen (haar, kleding, accessoires).
- Avatar-eigenschappen moeten instelbaar zijn via dropdowns en/of sliders.
- Wijzigingen moeten real-time zichtbaar zijn in een preview.
- Avatar-opties en kleuren moeten dynamisch geladen worden vanuit de API.
- De UI moet getest zijn op meerdere browsers en schermgroottes.

3.2 Functionele Requirements (Back-end)

- De backend moet avatar-instellingen per gebruiker kunnen opslaan en ophalen.
- Er moet een database-model bestaan voor avatar-instellingen.
- Avatar-data moet gevalideerd worden op geldige opties.
- De API moet avatar-wijzigingen correct opslaan en terugsturen naar de frontend.

4. Login & Authenticatie – Requirements

4.1 Functionele Requirements (Front-end)

- De applicatie moet een loginformulier bevatten.
- Gebruikersinvoer moet gevalideerd worden in de frontend.
- Bij foutieve inloggegevens moet duidelijke feedback worden getoond.
- Na succesvol inloggen moet een JWT-token of cookie opgeslagen worden.
- Het wachtwoordveld moet een zichtbaar/verbergen-toggle hebben.
- De loginpagina moet gestyled zijn volgens Figma.

4.2 Functionele Requirements (Back-end)

- De backend moet een login endpoint aanbieden (ExpressJS).
- Wachtwoorden moeten gehasht worden opgeslagen (bcrypt).
- Bij succesvol inloggen moet een JWT-token en/of cookie worden gegenereerd.
- De backend moet middleware bevatten voor beveiligde routes.
- Authenticatiegegevens moeten veilig worden teruggestuurd naar de frontend.

4.3 Niet-functionele Requirements (Security)

- Wachtwoorden mogen nooit in plaintext worden opgeslagen.
- Beveiligde routes mogen alleen toegankelijk zijn met een geldig token/cookie.

Tasks

Task	Assignee	Estimated hours	Spent hours	Progress
Schatkaart-pagina bijwerken zodat deze de actuele kalenderdata gebruikt (FE)	Rick	6	0	0%
Kalendercode opschonen (FE)	Rick	4	0	0%

Task	Assignee	Estimated hours	Spent hours	Progress
Componenten opsplitsen voor betere structuur (FE)	Rick	4	0	0%
Code duplicatie verminderen (FE)	Rick	3	0	0%
Front-end ↔ Back-end integratie (API connecties)	Rick, Boaz	8	4	50%
Weken- en maandenweergave implementeren	Rick	5	5	100%
Kalender met taken bouwen	Rick	6	6	100%
Taken aanmaken, bewerken en verwijderen (kalender)	Rick	6	6	100%
Drag & drop-systeem voor taken	Rick	8	8	100%
Taskpool maken voor taken buiten kalender	Rick	4	4	100%
Taken aanmaken, bewerken en verwijderen (taskpool)	Rick	5	5	100%
Kalenderpagina stylen volgens Figma	Rick	4	4	100%
Consistente UI-componenten toepassen	Rick	3	3	100%
Basis foutafhandeling en gebruikersfeedback	Rick	3	3	100%
API maken voor schatkaart-pagina (kalenderdata)	Boaz	5	0	0%
Taakdata ophalen en formatteren voor frontend	Boaz	4	0	0%
Backend endpoints opschonen en herstructureren	Boaz	5	0	0%
Data-validatie centraliseren (backend)	Boaz	3	0	0%
Duplicatie in databasequeries verminderen	Boaz	3	0	0%
API maakt kalenderdata beschikbaar voor schatkaart	Boaz	4	0	0%
Mapping controleren tussen taken en schatkaart	Boaz	3	0	0%
API CRUD taken kalender	Rick, Boaz	6	3	50%
Data beschikbaar stellen voor frontend	Rick, Boaz	4	2	50%
Endpoint testen met Postman	Rick, Boaz	3	1	30%
API voor taskpool CRUD-functionaliteit	Boaz	5	2	40%
Taskpool data opslaan en ophalen (backend)	Boaz	4	2	50%
Taken correct opslaan in database	Boaz	4	4	100%

Task	Assignee	Estimated hours	Spent hours	Progress
Avatar onderdelen toevoegen in UI	Valentijn	4	4	100%
Dropdowns / sliders voor avatar eigenschappen	Valentijn	3	3	100%
Real-time preview avatar wijzigingen	Valentijn	5	5	100%
Avatar kleuren en opties laden uit API	Valentijn	4	4	100%
Avatar UI testen (browsers/schermgroottes)	Valentijn	2	2	100%
Avatar customisatiepagina volgens Figma	Valentijn	4	4	100%
Interactieve selectie avatar onderdelen	Valentijn	4	4	100%
Avatar imports herstellen naar lib pad	Xander	2	2	100%
Layering van avatar onderdelen fixen	Xander	2	2	100%
Positionering avataronderdelen fixen	Xander	2	2	100%
API endpoint avatar ophalen	Xander	3	3	100%
Database-model avatar instellingen	Xander	4	4	100%
Validatie avatar data	Xander	3	0	100%
Avatar opslaan in database	Xander	4	4	100%
API retourneert avatar data naar frontend	Xander	3	3	100%
API endpoint avatar opslaan/bijwerken	Xander	4	4	100%
Prisma client genereren	Xander	1	1	100%
Database syncroniseren aan prisma	Xander	1	1	100%
Docker configuratie voor database	Xander	2	3	100%
Gateway koppelen aan avatar service	Xander	3	4	100%
Omgevingsvariabelen voor lokale db	Xander	1	1	100%
Db tabellen updaten via prisma push	Xander	2	2	100%
Login-formulier component maken	Bart	3	3	100%
Validatie logingegevens frontend	Bart	2	2	100%
Feedback tonen bij verkeerde login	Bart	2	1	50%
Login-formulier gestyled volgens Figma	Bart	3	3	100%
Wachtwoord zichtbaar/verborgen toggle	Bart	2	2	100%
Backend login endpoint (ExpressJS)	Bart	4	4	100%
Hashing wachtwoorden (bcrypt)	Bart	5	5	100%
JWT token genereren	Bart	3	3	100%
Middleware voor beveiligde routes	Bart	4	4	100%
Succesvol inloggen via backend	Bart	3	3	100%
Optionele cookie in plaats van JWT token aanmaken	Bart	5	5	100%

Retrospective summary

Start	Betere communicatie tussen teamleden Betere documentatie van werk en voortgang

	Betere samenwerking en kennisdeling tussen teamgenoten
Stop	Onduidelijke of incomplete updates in de planning Taken halverwege overnemen zonder afstemming Vertragingen door gebrek aan prioriteiten
Keep	Teamleden helpen elkaar bij moeilijke taken Front-end en Back-end taken goed gescheiden en toegewezen

Test (for sprint 2 and 3 only)

Summary of the tests

User Test 1 – Kalender: taken aanmaken

Start
- Mogelijkheid om taken snel toe te voegen via één duidelijke knop

Stop
- Te veel invoervelden bij het aanmaken van een taak

Keep
- Taken verschijnen direct zichtbaar in de kalender

User Test 2 – Task board: taken beheren

Start
- Drag & drop gebruiken om taken te verplaatsen

Stop
- Onduidelijke feedback na het verplaatsen van een taak

Keep
- Bewerken en verwijderen van taken werkt intuïtief

User Test 3 – Avatar: aanpassen van uiterlijk

Start
- Meer avatar-opties (bijv. haar, kleding, kleuren)

Stop
- Automatisch resetten van avatar-instellingen

Keep
- Directe visuele update bij het aanpassen van de avatar

User Test 4 – Avatar: opslaan en laden

Start
- Bevestiging tonen wanneer avatar succesvol is opgeslagen

Stop
- Avatar opnieuw moeten instellen na opnieuw inloggen

Keep
- Avatar blijft gekoppeld aan het gebruikersaccount

User Test 5 – Login

Start
- Duidelijke foutmelding bij verkeerde inloggegevens

Stop
- Onnodige stappen tijdens het inloggen

Keep
- Snel en eenvoudig inlogproces

Demo

Summary of the demo:
Tijdens de demo zijn de kalender, avatar-customisatie en login getoond. We lieten zien hoe taken kunnen worden toegevoegd, verplaatst en verwijderd, hoe avatars aangepast kunnen worden, en dat gebruikers kunnen inloggen en uitloggen.

Start
- Drag & drop van taken in de kalender werkte soepel
- De kalender toont taken direct na aanmaken
- Avatar kan aangepast worden met verschillende opties
- Snel en duidelijk inlogproces

Stop
- Avatar reset soms na refresh
- Login geeft soms vage foutmeldingen
- Te veel stappen bij het aanmaken van een taak

Keep
- Taken verschijnen direct en correct op de kalender
- Avatar-aanpassingen worden direct zichtbaar
- Login is overzichtelijk en eenvoudig te gebruiken

Design documentation

Use Case Specification sprint 3

Use Case: Taken aanmaken in de kalender

Use case naam:
Taken aanmaken in de kalender

Actor(s):
- Gebruiker
- Systeem (backend / database)

Summary description:
Een gebruiker kan een nieuwe taak aanmaken in de kalender. De taak wordt opgeslagen in de backend en direct zichtbaar op het kalenderbord.

Priority:
Hoog

Status:
Afgerond

Pre-condition:
- Gebruiker is ingelogd
- Kalenderpagina is geopend

Post-condition:
- Nieuwe taak is opgeslagen in de database
- Taak wordt zichtbaar op de kalenderpagina

Basic path:
1. Gebruiker klikt op “Nieuwe taak”
2. Gebruiker vult taaknaam, datum, beschrijving, optionele sub-taken en voegt personen toe aan de taak
3. Gebruiker klikt op “Opslaan”
4. Systeem slaat de taak op in de database
5. Taak verschijnt op de kalender

Alternative paths:
- Gebruiker vult een lege taaknaam in → Default taaknaam “taak” wordt ingevuld en de taak wordt opgeslagen
- Backend is niet bereikbaar → Systeem toont foutmelding en slaat taak niet op

Business rules:
- Elke taak moet minimaal een naam, datum en tijd hebben
- Een taak mag max 3 sub taken hebben

Non-functional requirements:
- Responsieve UI (werkt op mobiel en desktop)
- Opslaan van een taak mag maximaal 5 seconden duren
- Validatie en foutmeldingen duidelijk en begrijpelijk
- Backend moet minimaal 100 gelijktijdige gebruikers ondersteunen

Use Case 2: Avatar Customisatie

Use case name:
Avatar aanpassen

Actor(s):
- Gebruiker
- Systeem (backend / database)

Summary description:
Een gebruiker kan zijn avatar aanpassen door onderdelen zoals haar, kleding en accessoires te kiezen. De wijzigingen worden opgeslagen in de backend.

Priority:
Hoog

Status:
Afgerond

Pre-condition:
- Gebruiker is ingelogd
- Avatar-pagina is geopend

Post-condition:
- Avatar-instellingen zijn opgeslagen in de database
- Avatar wordt correct weergegeven bij volgende login

Basic path:
1. Gebruiker opent de avatar-pagina
2. Gebruiker selecteert onderdelen (haar, kleding, kleuren)
3. Gebruiker klikt op “Opslaan”
4. Systeem valideert en slaat wijzigingen op
5. Avatar wordt direct bijgewerkt op de pagina

Alternative paths:
- Gebruiker sluit pagina zonder opslaan → wijzigingen worden niet opgeslagen
- Backend niet bereikbaar → foutmelding tonen en opslaan blokkeren

Business rules:
- Avatar moet altijd een geldig uiterlijk hebben (minimaal één haar- en kledingoptie)
- Gebruiker kan avatar onbeperkt aanpassen

Non-functional requirements:
- UI moet direct de wijzigingen tonen (real-time preview)
- Opslaan mag maximaal 2 seconden duren
- Compatibel met mobiel en desktop

Use Case 3: Login

Use case name:
Gebruiker inloggen

Actor(s):
- Gebruiker
- Systeem (backend / authentication service)

Summary description:
Een gebruiker logt in met zijn gebruikersnaam en wachtwoord om toegang te krijgen tot de applicatie. Het systeem valideert de gegevens en start een sessie.

Priority:
Hoog

Status:
Afgerond

Pre-condition:
- Gebruiker heeft een account
- Applicatie is geopend

Post-condition:
- Gebruiker is ingelogd en kan alle functies gebruiken
- Sessie wordt gestart en beveiligd

Basic path:
1. Gebruiker opent het login-formulier
2. Gebruiker vult gebruikersnaam en wachtwoord in
3. Gebruiker klikt op “Inloggen”
4. Systeem valideert de gegevens
5. Systeem start sessie en navigeert naar de startpagina

Alternative paths:
- Onjuist wachtwoord → foutmelding tonen, gebruiker kan opnieuw proberen
- Backend niet bereikbaar → foutmelding tonen, login tijdelijk niet mogelijk

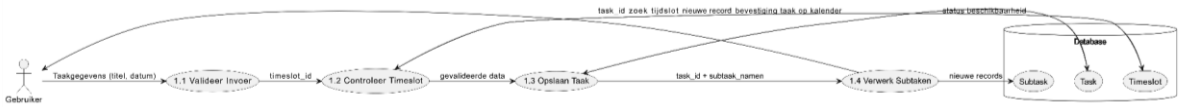
Business rules:
- Wachtwoord moet minimaal 8 tekens bevatten

Non-functional requirements:
- Loginproces mag maximaal 5 seconden duren
- Foutmeldingen duidelijk en begrijpelijk
- Systeem moet 1000 gelijktijdige logins ondersteunen

Dataflow Diagram per use case

Dataflow Diagram per use case
Use Case 1: Taken aanmaken in de kalender

In dit proces wordt een nieuwe taak vanuit de interface verwerkt en opgesplitst in de database.



Proces 1.1: Valideer Invoer

- Input: Taaknaam, datum en beschrijving (van Gebruiker).
- Output: Gecontroleerde data naar proces 1.2.
- Beschrijving: Het systeem controleert of de verplichte velden (titel, datum) zijn ingevuld volgens de business rules.

Proces 1.2: Controleer Timeslot

- Input: timeslot_id (van proces 1.1).
- Output: Status van de beschikbaarheid (uit tabel Timeslot).

- Beschrijving: Het systeem verifieert in de database of het gekozen tijdslot (bijv. ochtend of middag) bestaat en gekoppeld kan worden.

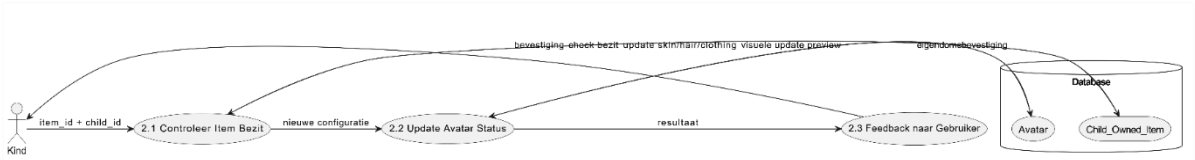
Proces 1.3: Opslaan Hoofdtuaktaak

- Input: Gevalideerde taakset.
- Output: Nieuw record in tabel Task; genereren van task_id.
- Beschrijving: De hoofdtaak wordt weggeschreven naar de database. Het gegenereerde ID is nodig voor de volgende stap.

Proces 1.4: Verwerk Subtaken

- Input: Subtaak-namen en task_id (van proces 1.3).
- Output: Nieuwe records in tabel Subtask.
- Beschrijving: Eventuele subtaken worden gekoppeld aan de hoofdtaak en opgeslagen in de database (maximaal 3).

Use Case 2: Avatar Customisatie



Dit proces beschrijft hoe een kind het uiterlijk van zijn of haar avatar aanpast.

Proces 2.1: Controleer Item Bezit

- Input: child_id en item_id (van Kind).
- Output: Eigendomsbevestiging (uit tabel Child_Owned_Item).
- Beschrijving: Voordat een item (zoals kleding) wordt toegepast, controleert het systeem of het kind dit item daadwerkelijk in bezit heeft.

Proces 2.2: Update Avatar Status

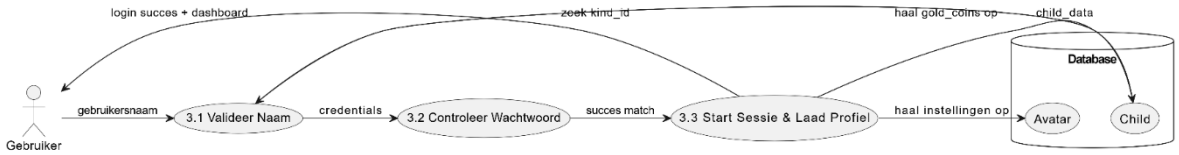
- Input: Nieuwe configuratie (haarstijl, kleur, kleding_id).
- Output: Bijgewerkt record in tabel Avatar.
- Beschrijving: De nieuwe uiterlijke kenmerken worden opgeslagen in de database onder het profiel van het kind.

Proces 2.3: Feedback naar Gebruiker

- Input: Bevestiging van de database.
- Output: Real-time visuele update.

- Beschrijving: De interface krijgt een seintje dat de wijziging is geslaagd, waardoor de preview van de avatar direct wordt bijgewerkt.

Use Case 3: Gebruiker inloggen



Dit proces regelt de toegang tot de applicatie en het laden van de persoonlijke omgeving.

Proces 3.1: Valideer Identiteit

- Input: Gebruikersnaam (van Gebruiker).
- Output: Child_data (uit tabel Child).
- Beschrijving: Het systeem zoekt de naam op in de database om te kijken of er een bijbehorend kind-account bestaat.

Proces 3.2: Controleer Gegevens

- Input: Credentials en wachtwoord.
- Output: Validatie resultaat.
- Beschrijving: De ingevoerde gegevens worden gecontroleerd door de authenticatie-service.

Proces 3.3: Start Sessie & Laad Profiel

- Input: Bevestigd child_id.
- Output: Saldo (gold_coins) en Avatar-instellingen naar de gebruiker.
- Beschrijving: Na een succesvolle login worden de specifieke gegevens van het kind geladen, zodat het dashboard gepersonaliseerd getoond kan worden.

API specification and Message(s)

API Name: Api-Gateway

Version: 4.0

Base-Url: http://Localhost:3012/

Authentication: Bearer Token (JWT)
Header: Authorization: Bearer <access_token>
Applied Via: authenticateToken

Routes Overview:

Routes Overview

Route	Method	Auth Required	Description
`/`	GET	❌	Health check
`/auth/register`	POST	❌	Register a new user
`/auth/login`	POST	❌	Authenticate user
`/auth/me`	GET	✅	Get current user
`/auth/logout`	POST	❌	Logout user
`/owners/*`	ALL	✅	Proxy to Client service
`/appointments/*`	ALL	❌	Proxy to Appointment service
`/timeslots/*`	ALL	✅	Proxy to Timeslot service
`/avatar/*`	ALL	❌	Proxy to Avatar service (temporary)

Release

Final release for Front-end:
https://github.com/rickdecuijper/UVC-Project-Front-End/releases/tag/v1

Final release for Back-end:
https://github.com/rickdecuijper/UVC-project-Back-End/releases/tag/v1.1

Pull request

Every student should code. In this table you relate an issue on Github with a task you mention above. When finishing an issue is a pull request should be created and one of the other students should review the pull request and either reject or accept the request. Every sprint a students should at least make to pull requests.

Front-end pull requests

Nr	Issue	Task	Responsible	Reviewed
29	Feature/calendar components	Kalender componenten toegevoegd	Rick	Approved
28	Avatar: frontend ↔ backend	Avatar frontend verbonden aan backend en component gemaakt	Xander	Approved
27	Avatar systeem in dev branch	Avatar systeem toegevoegd in dev branch	Valentijn	Approved
26	delete +page.js	Verwijderd +page.js	Valentijn	
25	Front-end kalender af	Kalender functionaliteit op frontend afgerond	Rick	
24	Revert "Feature/frontend fix"	Revert commit, review nodig	Rick	Review required
23	Feature/frontend fix	Frontend fix doorgevoerd	Valentijn	Approved

22	Schatkaart pagina	Schatkaart pagina gemaakt en gestyled, achtergrond-uploader toegevoegd aan kalenderpagina	Xander	Approved
21	Kalender pagina+styling	Kalenderpagina gemaakt en gestyled	Xander	Approved
20	Kalender pagina+styling	Kalenderpagina styling (oude PR gesloten)	Xander	Approved
18	Update dev (#17)	Dev branch geüpdatet	Rick	Review required
17	Update dev	Update dev branch	Rick	
16	Avatar systeem	Avatar systeem PR (gesloten)	Valentijn	Changes requested
15	Update node.js.yml	Node.js pipeline update	Rick	Approved
14	Update route.test.js	Route unit test geüpdatet	Rick	Approved
13	Update dev branch (#12)	Dev branch geüpdatet	Rick	Review required
12	Update dev branch	Dev branch update	Rick	
10	Hotfix unit tests	Hotfix voor unit tests	Rick	Approved
9	Kalender pagina styling	Kalenderpagina gemaakt met styling (gesloten)	Xander	Changes requested
31	release dev to main	*No description provided.*	All teammembers	Approved

Back-end pull requests

Nr	Issue	Task	Responsible	Reviewed
38	Removed outdated files	Verwijderd van verouderde bestanden	BoazVaneveld	Approved
37	Database backend	Eerste versies van database toegevoegd aan backend	rickdecuijper / BoazVaneveld	Approved
36	Revert database backend	Revert van vorige database commit	rickdecuijper	Approved
35	Database backend	Eerste versies van database toegevoegd aan backend	BoazVaneveld	Approved
34	Avatar backend	Backend logica voor avatar systeem toegevoegd	Xanderu07	Approved
33	Login	Login-functionaliteit toegevoegd	bjhgeijtenbeek	Approved
39	release to main	*No description provided.*	All teammembers	Approved