1. Tokens: de authenticatie:

Onze authenticatie verloopt in enkele stappen:  
stap 1) is de user logged intern in Laravel?  
ja => geauthenticeerd => ga door naar authorisatie  
neen=> ga naar stap 2  
stap 2) heeft de user een token meegestuurd als parameter?  
ja => is deze token geldig voor een user?  
 ja=> log deze user in en ga door naar authorisatie

(deze code is onder middleware CheckToken te vinden)

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving  
hier kijken we of de request de token als input parameter heeft, zoja dan kijken we of er een user is met de waarde van die token in onze database. Zodien er een user is (en dus niet gelijk aan null) dan wordt die user ingelogd, wordt de sessie vernieuwd en gaan we door naar de volgende middleware checks. Als de user niet gevonden werd => null dan wordt er dus niemand ingelogd en gaan we door naar de volgende middelware.

Neen => ga naar stap 3  
stap 3) return json met error en bericht om in te loggen.

Stap 4) user stuurt post naar /api/v1/mapi\_login met “name” en “password” deze worden via de Auth facade nagekeken tegenover de hash in de database. Als deze auth attempt een succes was wordt de consumer gekozen uit de user repo, en met deze user wordt de functie createMApiToken aangeroepen. In deze functie wordt er gekeken naar welk type de consumer heeft. Dat word de “security level” van de consumer, hierna word afhankelijk van het security level bepaalde “security clearences” toegewezen aan de “security clearences array” de token krijgt een herkenbare naam door de consumer id en security level te combineren. Nu wordt gekeken of er een PErsonalAccesToken van Sanctum bestaat. Zoja dan zal de oude token eerst verwijderd worden. Eens de oude token al dan niet verwijderd is kan er een nieuwe token gemaakt worden met tokennaam, en security clearences mee opgegeven. De gegevens worden opgeslagen en in een array en als response gegenereerd. En terug gegeven aan de login functie. De login functie zal deze dan als JSON response geven. Als de attempt gefaald was dan wordt simpelweg een JSON teruggegeven met een 401

1. Gates (Authorization)

(in authserviceproviders)  
Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Deze Gates komen vanuit laravel zelf en dienen om ge-autenticeerde users na te kijken op een voorwaarde, Deze gates worden gebruikt om het type user na te kijken en een gate om na te kijken of de user die de aanvraag deed dezelfde user is op wie de aanvraag betrekking heeft.

Hieronder een voorbeeld hoe deze gates toegepas zijn in de code in UserController:

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Hier zijn de Gates geschreven zodat eerst nagekeken wordt of de user door een van de opgelijst gates geautoriseerd is. Als dit zo is faalt de “if niet” en gaat de code door, als de user dit faalt zorgt de niet(!) ervoor dat de if statement “True” wordt en de user zal een 403 “not authorized” redirect krijgen.

Bijvoorbeeld bij Show is de isCurrentConsumer gate actief, dit zorgt ervoor dat ook non-admins en non-robots dit opzichzelf kunnen aanroepen, maar niet op andere consumers.