BanKING

Created and documented by: Hugo Korte



Inhoud

[Backend: System Description 3](#_Toc106910439)

[About: 3](#_Toc106910440)

[Technical Description: 3](#_Toc106910441)

[Controllers: 3](#_Toc106910442)

[Services: 3](#_Toc106910443)

[Repositories: 3](#_Toc106910444)

[Backend: Class Diagrams 4](#_Toc106910445)

[Class Diagram: BanKING 4](#_Toc106910446)

[AppUser: 4](#_Toc106910447)

[BankAccount: 4](#_Toc106910448)

[Transaction: 4](#_Toc106910449)

[Backend: Rest Specification 5](#_Toc106910450)

[Access-token controller: 5](#_Toc106910451)

[App-user controller: 5](#_Toc106910452)

[Transaction controller: 8](#_Toc106910453)

[Backend: Sequence Diagrams 10](#_Toc106910454)

[Sequence: Sign Up 10](#_Toc106910455)

[Sequence: Deposit 11](#_Toc106910456)

[Backend: Implementation Choices 12](#_Toc106910457)

[JWT: 12](#_Toc106910458)

[Lombok; 12](#_Toc106910459)

[Gson: 13](#_Toc106910460)

[Spring Security 13](#_Toc106910461)

[Backend: Test Rapport 14](#_Toc106910462)

[Frontend: Functionality Description 15](#_Toc106910463)

[Frontend: Use Case Diagram 16](#_Toc106910464)

[Frontend: Wireframes 17](#_Toc106910465)

[Home page: 17](#_Toc106910466)

[Sign-up page: 17](#_Toc106910467)

[Sign-in-page: 18](#_Toc106910468)

[Account overview page: 18](#_Toc106910469)

[Justification of other choices: 19](#_Toc106910470)

# Backend: System Description

## About:

BanKING is een full-stack paypal-achtige online bank applicatie die bedoeld is om gebruikers hun geld veilig te kunnen laten opbergen, ophalen en versturen. Ook kun je alle transacties die in het verleden zijn gemaakt weergeven, verwijderen en natuurlijk nieuwe maken. Het weergeven gebruikt ook nog verschillende sorteer algoritmes om de transacties overzichtelijker te maken.

## Technical Description:

### Controllers:

De controllers zijn het start en eindpunt van elke request, deze controllers hebben als functie om alle requests op te vangen en als response een van de functies in de service klassen aan te roepen die dan de request kan verwerken en bijbehorende response kan genereren.

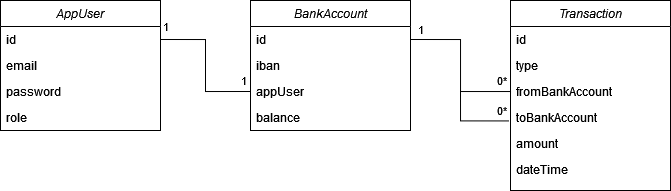
Services:  
De services zijn de klassen die alle ‘Business logic’ hanteren. De services hebben meestal een aantal methodes die relevant zijn aan de service en nemen een request als input (met soms parameters) en die dit request verwerken tot een netjes pakketje die ze dan weer terugsturen naar de controller.

### Repositories:

De repositories dienen als tussen-klassen waardoor de services met de database kan communiceren. Dit betekent dus dat de repositories een aantal methoden hebben (denk aan: Find By, Save en Delete) en dat die methoden worden omgezet in SQL statements waardoor de CRUD functionaliteit mogelijk is op de database.

# Backend: Class Diagrams

## Class Diagram: BanKING



Het klasse diagram bestaat uit 3 klassen:

1. *AppUser:*
2. *BankAccount*
3. *Transaction*

### AppUser:

De *AppUser-*klasserepresenteert een gebruiker van de applicatie.

### BankAccount:

De *BankAccount-*klasse representeert een bank rekening en heeft een verplichte One-To-One relatie met de *AppUser* (dit kun je zien als de eigenaar van de bank rekening). Wanneer een *AppUser* wordt aangemaakt wordt dan ook al direct een *BankAccount* aangemaakt en gekoppeld.

### Transaction:

De *Transaction*-klasse representeert een transactie tussen 2 *BankAccount-*klassen en heeft dus een One-To-Many relatie met een *BankAccount-*klasse, omdat een transactie (vaak) tussen 2 bank rekeningen is kan 1 transactie 2 relaties hebben met 2 verschillende *BankAccount-*klassen.

# Backend: Rest Specification

## Access-token controller:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/access-token** | | | |
| [description here. Explain what the request does when executed successfully.] | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | refresh-token | auth header | jwt waarmee je de access-token mee kan refreshen |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | access token + access token is opgehaald | |
| 422 | de refresh token was niet geldig | |
|  |  | |
|  |  | |
|  |  | |

## App-user controller:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **/api/app-users** | | | |
| Dit request zal een app-user aanmaken en registreren in de database | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | app-user\* | body | Email en wachtwoord van app-user |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 201 | app-user is aangemaakt | |
| 409 | app-user met die email bestaat al | |
| 422 | De request mist/bevat verkeerde informatie | |
|  |  | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/app-users/{email}** | | | |
| [description here. Explain what the request does when executed successfully.] | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | app-user\* | path | Email van app-user die opgehaald moet worden |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | app-user was opgehaald | |
| 401 | request heeft niet benodigde rechten om app-user op te halen | |
| 404 | app-user was niet gevonden | |
|  |  | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PUT** | **/api/app-users/{email}** | | | |
| Update app-user password by specifying email from app-user | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | email\* | path | app-user dat veranderd moet worden |
| Wachtwoord\* | body | wachtwoord dat veranderd moet worden |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 204 | app-user wachtwoord is veranderd | |
| 401 | request heeft niet benodigde rechten om app-user te veranderen | |
| 404 | app-user met die email kon niet worden gevonden | |
| 422 | wachtwoord voldoet niet tot criteria | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **/api/app-users/{email}** | | | |
| Delete app-user by specifying email from app-user | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | email\* | path | email van app-user die verwijderd moet worden |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 204 | app-user was verwijderd | |
| 401 | geen toegang tot deze resource | |
| 404 | email was niet gevonden | |
|  |  | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **/api/app-user/authenticate** | | | |
| Authenticate user and get JSON Web Tokens (JWT) to access any other resources without re-authenticing | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | email\* | body | email van app-user |
| password\* | body | wachtwoord van app-user |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | response json bevat access en refresh token | |
| 401 | incorrect wachtwoord | |
| 404 | email was niet gevonden | |
|  |  | |
|  |  | |

## Transaction controller:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| POST | **/api/bank-account/{iban}/transactions/{TransactionType}** | | | |
| Create a transaction that allows the user to alter the balance of their bankaccount | | | | |
|  | | | | |
| Parameters: | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | iban\* | path | email van app-user |
| transactietype\* | path | type transactie (storten, ophalen, etc) |
| amount\* | body | Aantal euro’s |
| bankAccTo | body | iban waar verzend requests heen gaan |
|  |  |  |
| Responses: | | **Code** | **Description / example if successful** | |
|  | | 201 | transactie was successvol | |
| 401 | geen toegang tot deze resource | |
| 404 | iban was niet gevonden | |
| 422 | aantal of iban is incorrect | |
|  |  | |

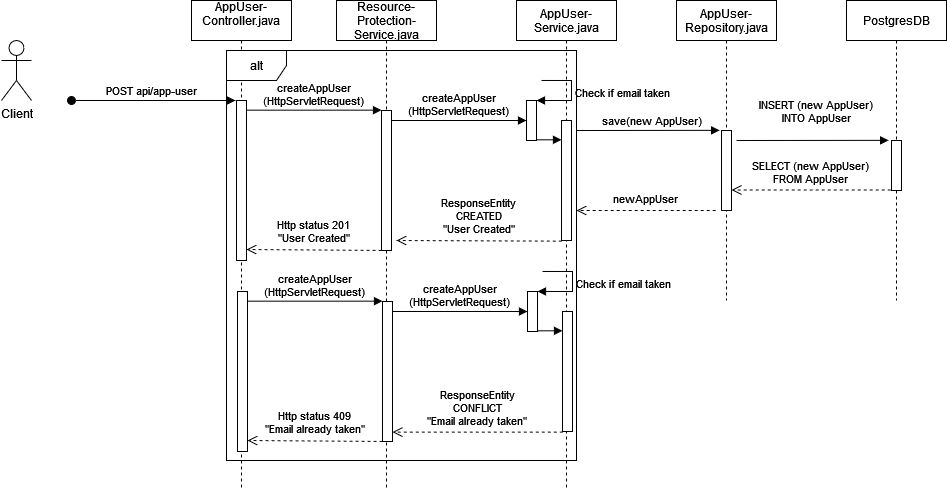
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/bank-account /{iban}/transactions?limit={limit}&sortBy={sortBy]** | | | |
| [description here. Explain what the request does when executed successfully.] | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | iban\* | path | transactions of iban |
| limit | query | aantal transacties |
| sortBy | query | sorteer algoritme |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | json met alle transactie’s | |
| 401 | geen toegang tot deze resource | |
| 404 | iban niet gevonden | |
|  |  | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PUT** | **/api/bank-account/{iban}/transaction/{id}** | | | |
| Update app-user password by specifying email from app-user | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | iban\* | path | app-user dat veranderd moet worden |
| id\* | path | wachtwoord dat veranderd moet worden |
| fromBankAcc\* | body | bank rekening van zender |
| toBankAcc\* | body | bank rekening van ontvanger |
| amount\* | body | aantal euro’s |
|  | | timestamp\* | body | tijd en datum |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 204 | transactie is veranderd | |
| 401 | geen toegang tot deze resource | |
| 404 | transactie nog niet gevonden | |
| 422 | zender/ontvanger/aantal/tijdendatum was niet goed | |
|  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **/api/app-user/{iban}/transactions** | | | |
| [description here. Explain what the request does when executed successfully.] | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *Add a \* to the name of required parameters.* | | iban\* | path | iban van bankrekening |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 204 | transacties verwijderd succesvol | |
| 401 | geen toegang tot deze resource | |
| 404 |  | |
|  |  | |
|  |  | |

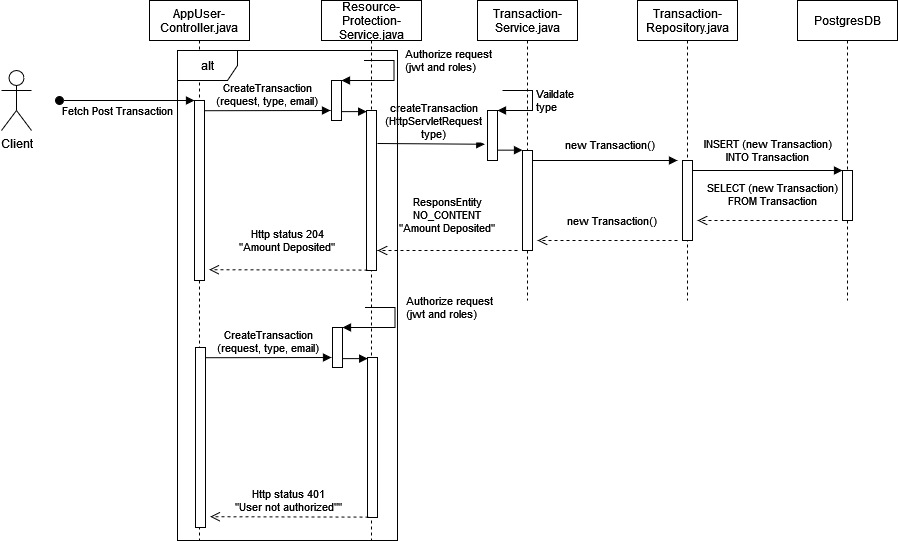
# Backend: Sequence Diagrams

## Sequence: Sign Up



In dit sequence diagram wordt gevisualiseerd hoe een gebruiker een account aan kan maken (sign-up) met mijn applicatie. Om het proces goed te visualiseren is bij elke laag neergezet hoe ze de request verwerken en wat ze als response teruggeven. Voor elk request dat binnenkomt bij AppUser-Service.java word een validatie check gedaan zodat alleen geldige email adressen zich kunnen aanmelden.

## Sequence: Deposit



In dit sequence diagram wordt gevisualiseerd hoe een gebruiker geld kan storten op zijn/haar bank rekening via Deposit. Wanneer een request aankomt in de ‘ResourceProtectionService.java’, word elke request, als dat nodig is, geauthoriseert door middel van de rol uit de access token te halen en kijken of die rol de benodigde ‘LevelOfClearance’ bevat om toegang tot die resource te krijgen. Hier wordt ook bekeken wanneer de rol USER is dat de access token en de resource dezelfde eigenaar hebben zodat users alleen bij hun eigen bankaccount en transacties kunnen.

# Backend: Implementation Choices

In de BanKING applicatie worden meerdere dependencies om de developement tijd en applicatie te verbeteren, voor elke dependency word uitgelegd *wat* het is en  *waarom* BanKING dit gebruikt.

## JWT:

JWT (ja-wt) staat voor Json Web Token en is een van meerdere manieren om authentication en authorization te behandelen. Een JWT is een lange set aan karakters die uit 3 gedeelte is gesplitst door middel van punten. Deze 3 gedeeltes zijn:

1. Header

De Header bevat informatie over de JWT, dit is meestal het algoritme waarmee de token is gesigned.

2. Payload

De Payload bevat informatie wat de token moet behouden, dit is vaak een gebruiker (subject), de rollen van die gebruiker (claims) en de datum wanneer de token is gemaakt (dit is nodig om te weten wanneer de token ongeldig wordt verklaart).

3. Signature

De Signature is het laatste en ook belangrijkste gedeelte van JWT, de signature is een soort van check dat de inhoud van de token niet veranderd is. Meestal heeft de applicatie die de JWT’s signed een secret die wordt gebruikt met het algoritme om de set karakters te genereren, wanneer de JWT’s inhoud veranderd is zal de signature niet meer goed zijn en wanneer de secret wordt gebruikt om de JWT te decoden zal hier een inconsitentie ontstaan en de JWT ongeldig maken.

Dit wordt in BanKING gebruikt omdat het een goeie oplossing is om serversided-sessions tegen te gaan zodat de API RESTful blijft.

## Lombok;

Lombok is een Java bibliotheek die probeert het probleem van boilerplate aan te pakken door middel van annotations. Deze annotaties kunnen boven klassen worden aangeduid en maken (onder de motorkap) dus Java boilerplate zodat het schrijven van Java een stuk sneller gaat doordat bijvoorbeeld voor 10 variabelen een getter moet maken kun je boven aan in de klasse de annotatie: ‘Getter’ neerzetten en kun je dus getters gebruiken zonder ze te maken/zien. Dit zorgt dus voor minder boilerplate en schonere/leesbare code.

## Gson:

Gson is een library die wordt gebruikt voor het parsen en formatteren van JSON. Dit maakt het makkelijk om binnenkomende requests makkelijk uit elkaar te halen, en responses makkelijker te formatteren.

## Spring Security

De laatste library die wordt gebruikt in BanKING is Spring Security, spring security bied een hele set aan tools aan om een applicatie veiliger te maken (filters, configuratie’s, etc). BanKING gebruikt geen van deze features omdat dit simpelweg overkill was voor deze applicatie. De reden dat ik deze library nog wel benoem is omdat Spring Security BCryptPasswordEncoder bevat wat ik gebruik om de wachtwoorden van de gebruikers te Hashen, dit betekent dat het wachtwoord aan het BCrypt algoritme wordt gevoerd en dat daar een gefixeerde lengte aan karakters uitkomt waardoor zelfs BanKING de wachtwoorden van de gebruikers niet weet. Let op dit is hashing en niet encrypting, hashing is ‘one-way’. Dit betekent dat wanneer het wachtwoord gehashed word, niet meer terug kan worden gezet naar de orginele wachtwoord. Encryption is ‘two-way’ en zorgt ervoor dat wanneer een wachtwoord omgezet wordt (encrypt) dat het ook weer terug kan worden gezet naar het orginele wachtwoord (decrypt).

# Backend: Test Rapport

Om de backend te testen heb ik Postman gebruikt om alle requests naar de API te sturen, ook heb ik hierbij alle happy en unhappy -paths getest waardoor ik elke response status kon testen en verifiëren.

Hier is een foto van de postman collecties die ik bijhield om requests te verdelen in verschillende groepen (controllers) te delen en zo makkelijk identieke requests kon maken naar de API voor een consistent test resultaat.

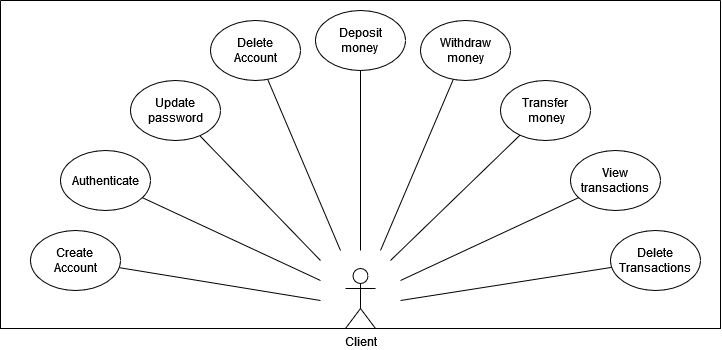
Afbeelding met tekst

Automatisch gegenereerde beschrijving

# Frontend: Functionality Description

De Frontend van BanKING is volledige geschreven in vanilla HTML/CSS/Javascript en is gemaakt om het communiceren met de API te versimpelen voor gebruikers. Om dit voor elkaar te krijgen zijn er nette en opgemaakte pagina’s gemaakt met feedback op elke actie zodat het ondernemen van acties op de API simpel en snel word. Om de front end te testen heb ik familie en vrienden gebruikt die ik mijn site liet gebruiken zodat zij functionele of visuele feedback konden geven op de UI. Dit was mogelijk dankzij GitHub Pages waardoor ik gratis mijn site (statisch) kon hosten, hierdoor heb ik ook met behulp van media query’s de website ook op apparaten zoals telefoons en iPad ’s kunnen ondersteunen.

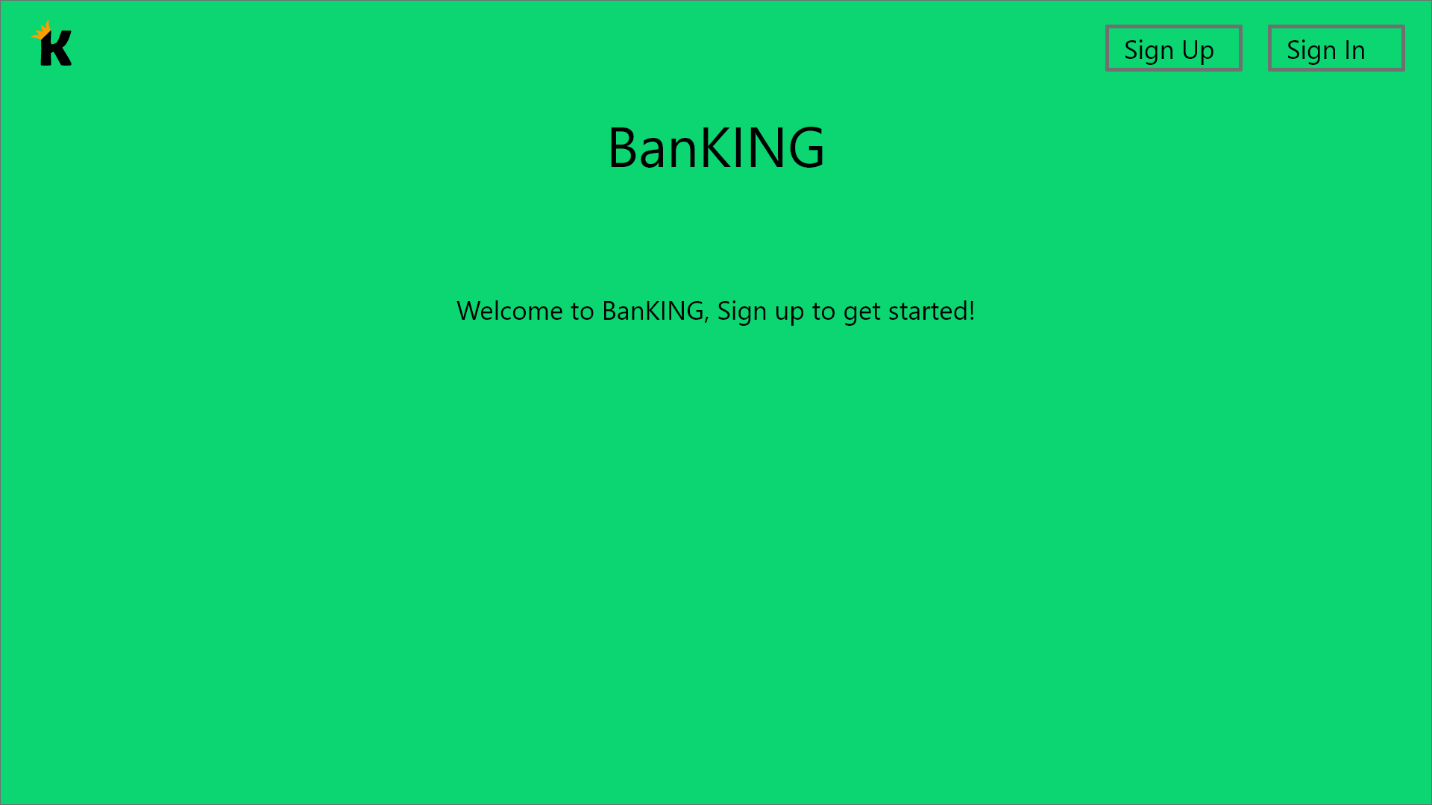
# Frontend: Use Case Diagram



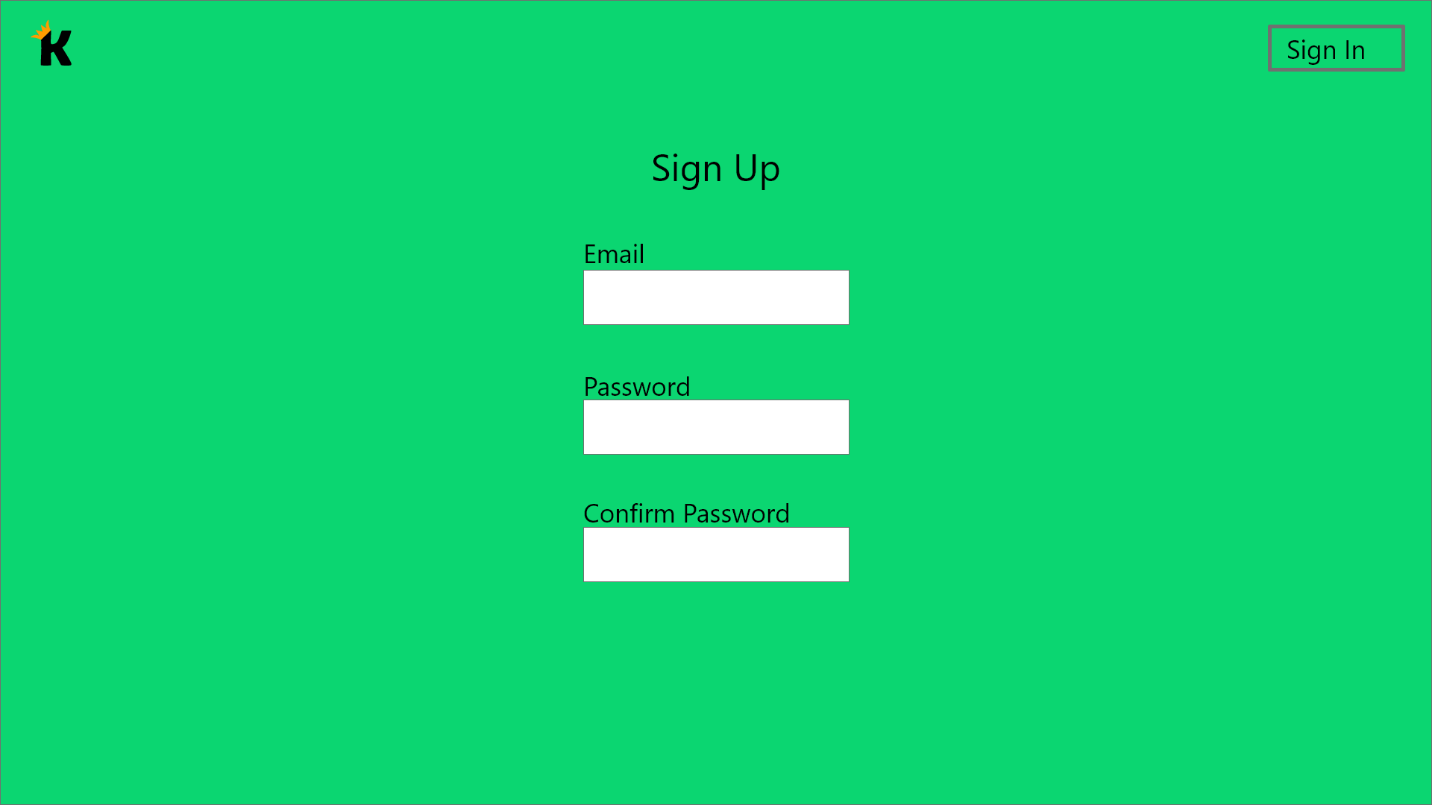
Dit diagram geeft aan wat een client allemaal kan doen met de API van BanKING.

# Frontend: Wireframes

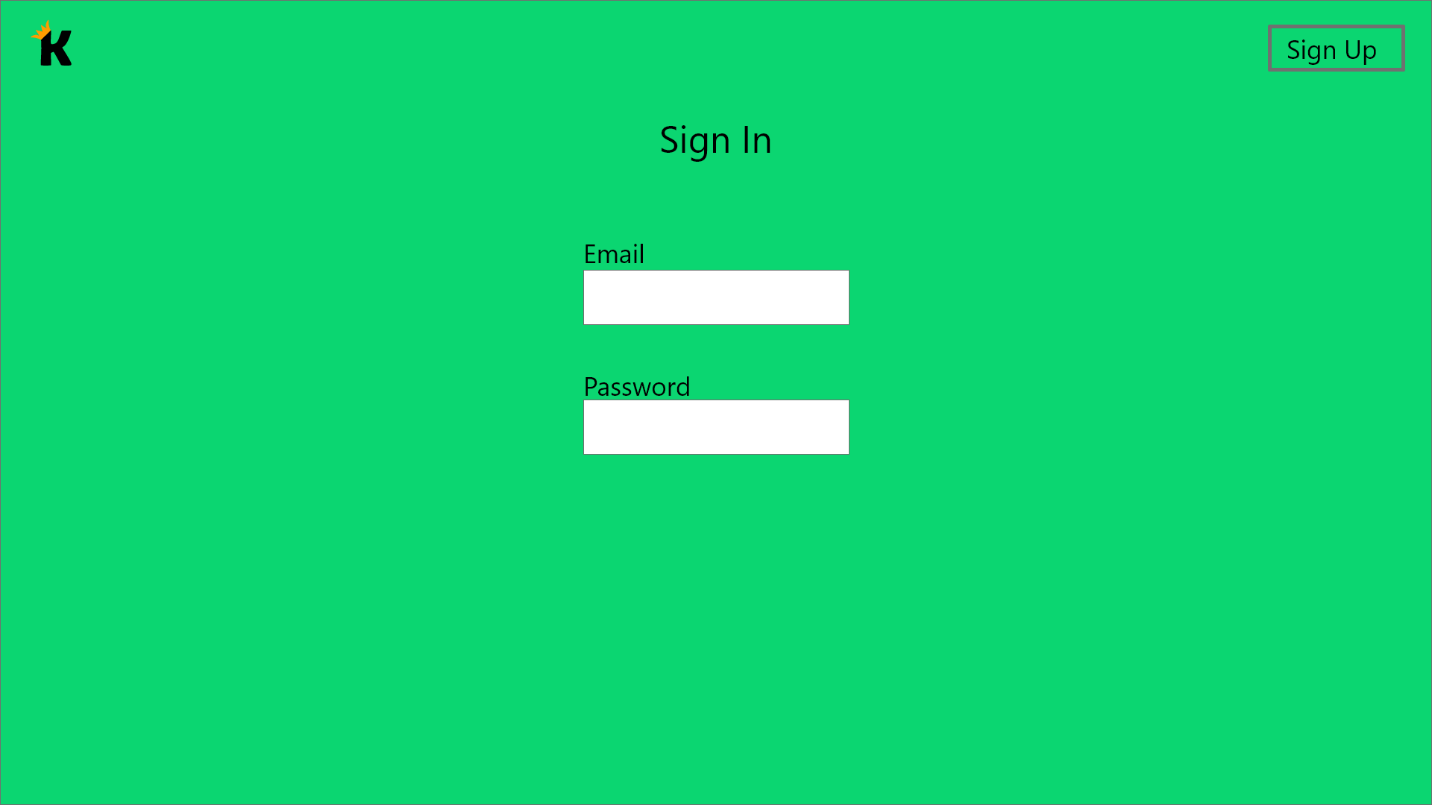
## Home page:



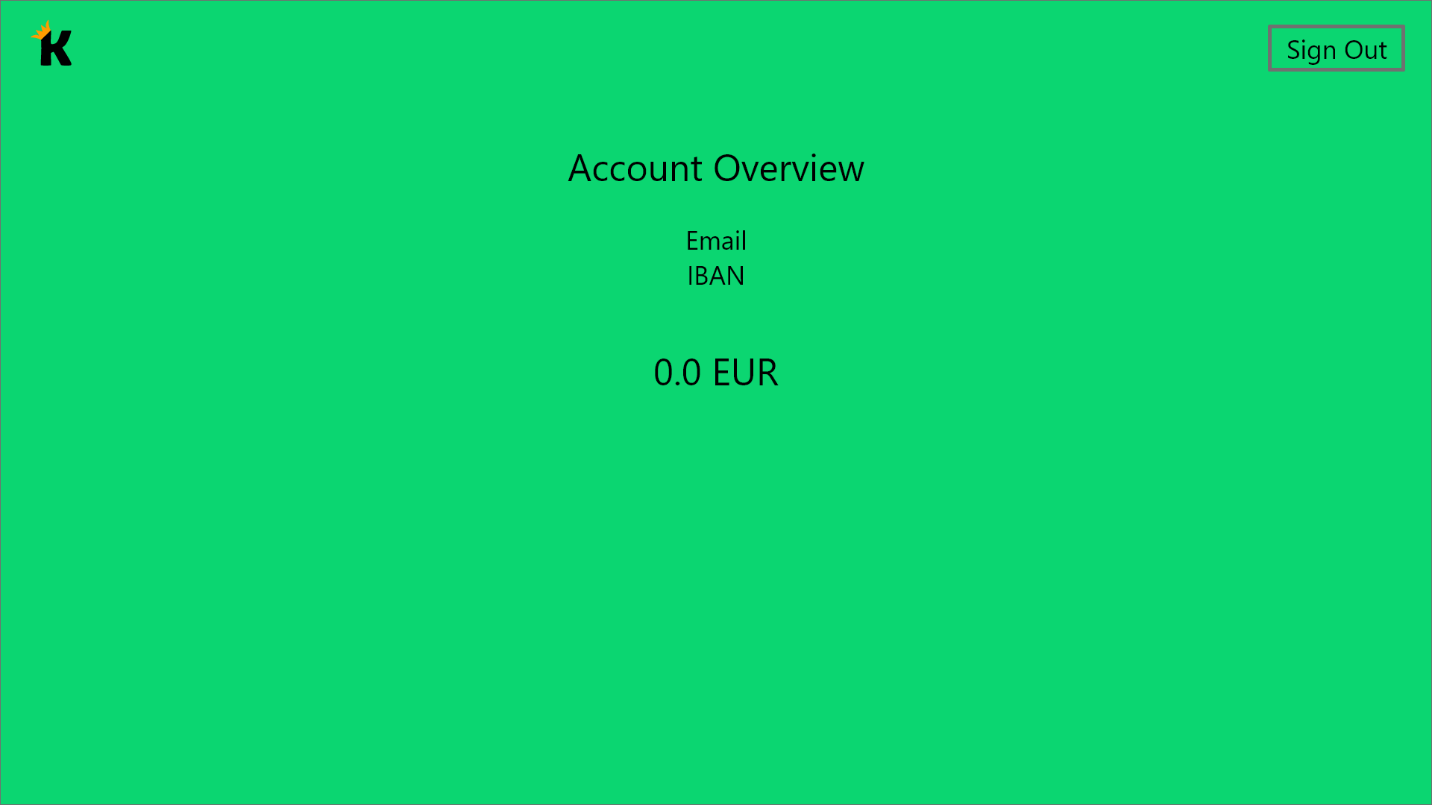
## Sign-up page:



## Sign-in-page:



## Account overview page:



# Justification of other choices:

Ik heb gekozen voor een simpele UI waarbij de kleuren veel contrast hebben vergeleken met elkaar zodat alles erg duidelijk zichtbaar is, ook heb ik animaties toegevoegd zodat de site iets minder statisch is. Ook is er een util.js aanwezig waarin wat hulpmethode ’s staan die gebruikt worden in meerdere scripts door middel van het exporteren/importen van constanten en functies, dit verminderd de hoeveelheid duplicaat code.