Design Document
Strippenkaart

Opdrachtnemer
Collin Franckena
227398
DEV3
Applicatie- & Mediaontwikkelaar
Friesland College, Heerenveen

Opdrachtgever
Jan Zuur
Friesland College

Document
25-02-2020
Versie: 0.1

Contents

1 Inleiding

Het doel van dit document is om de ontwikkeling en implementatie van de strippenkaart API, Android app en Web app te beschrijven. Het strippenkaart systeem is ontworpen om strippenkaarten te kunnen verkopen aan niet leden. Dit maakt het mogelijk om voor niet leden van een verenging toch mee te doen aan een activiteit van een strippenkaart zonder daar ter plekke te betalen.

Dit documnet is bedoeld als handleiding voor ontwikkelaars die met de Strippenkaart applicatie bezig gaan.

1.1 Versie geschiedenis

Versie	Datum	Status	Omschrijving
0.1	04-02-2020	Opzet	Eerste opzet design document

1.2 Distributie

Naam	Rol	Email
Collin Franckena	Student/Opdrachtnemer	*Collin.franckena001@fclive.nl*
Jan Zuur	Docent/Opdrachtgever	*zuja@fcroc.nl*

1.3 Term definities

Hieronder zijn de definiteies te vinden van een aantal termen in dit document.

Term	Definitie
API	*De API handelt alle gegevensverzoeken af van het systeem.*
	Dit houdt bijvoorbeeld in: ophalen/toevoegen/bewerken/verwijderen van gegevens.
Tikkie	*Een online service die het mogelijk maakt om betaalverzoeken te sturen naar klanten.*
QR-code	*Een type barcode die door de camera van een telefoon gescand kan worden.*
	Dit maakt het mogelijk om gegevens makkelijk uit te wisselen.
CRUD	*CRUD is een afkorting voor Create, Read, Update en Delete.*
Klasse/Class	*Een klasse is een blauwdruk van een object.*
Object	*Een object is een instantie van een klasse*

2 Project specificaties

2.1 Inleiding

De opdracht geconstrueerd door Jan Zuur en is op 4 februari 2020 aan mij gegeven. De systeem word gebouwd voor de vereniging Delta Impuls, maar moet ook bij andere verenigingen ingezet kunnen worden. De doelgroepen hierbij zijn *niet leden*, *penningmeeesters*, *beheerders* en *verengingen*.

2.2 Samenvatting

Het project is het maken van een systeem die het mogelijk maakt voor niet leden van een vereniging om makkelijker mee te doen met een activiteit van een vereniging. Met dit systeem kunnen niet leden een strippenkaart aanschaffen en dan bij die activiteiten betalen met hun strippenkaart, hierdoor hoeven de niet leden niet elke keer te betalen bij een activiteit. Dit maakt de drempel om mee te doen met een activiteit als niet lid lager.

Het systeem bestaat uit drie onderdelen: een *API*, deze verzorgt alle gegevens aan de andere twee onderdelen. een *mobiele app voor Android* en een *website (web app)*, deze twee onderdelen zullen dezelfde features hebben.

3 Actors

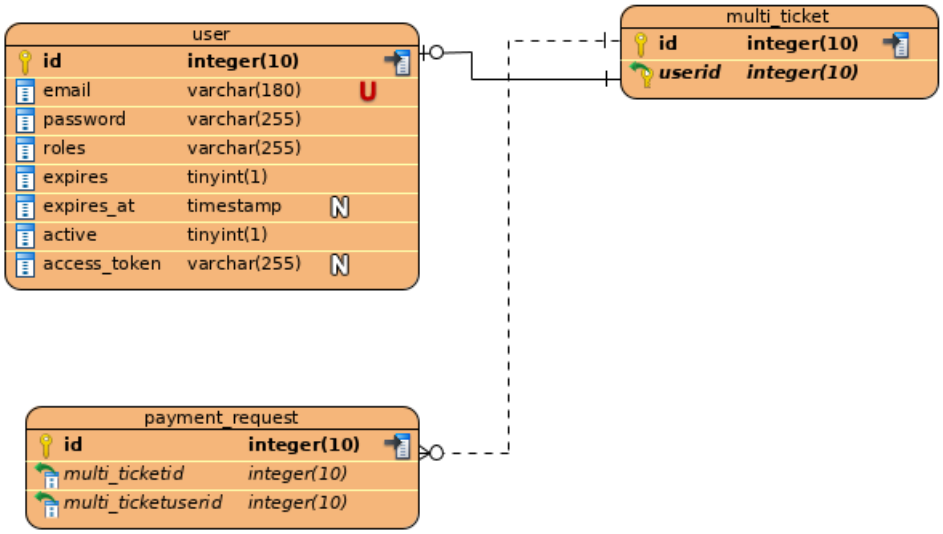
In dit hoofdstuk worden alle actoren beschreven. Alleen deze actoren kunnen interactie hebben met het systeem.

- **Niet leden** - Dit zijn de mensen die niet direct lid zijn van een vereniging. Deze mensen zijn het hoofddoel van dit project. Deze niet leden moeten strippenkaarten kunnen aanschaffen en met deze strippenkaart kunnen betalen bij een activiteit.
- **Beheerders** - De beheerders zijn de coaches/trainers van een vereniging. De groep moet de strippen van een strippenkaart kunnen strippen.
- **Penningmeesters** - De penningmeesters zijn de personen die het geld beheren van een vereniging. De penningmeester moet de betaalverzoeken kunnen inzien en ook Tikkies kunnen versturen.

4 Diagrams

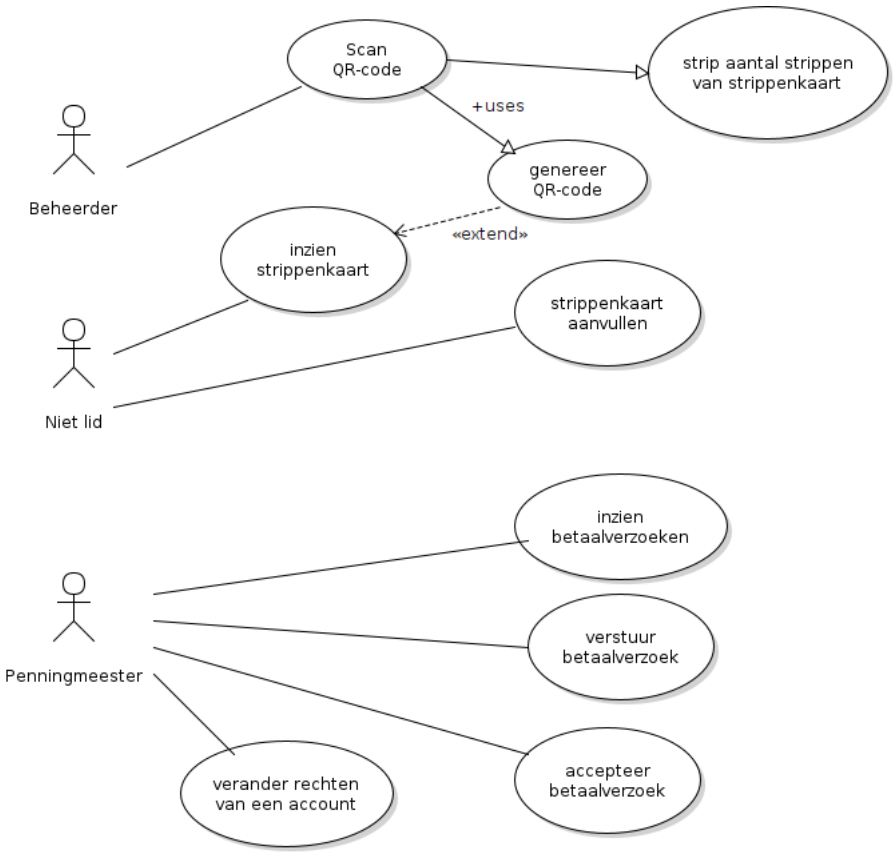
In dit hoofdstuk worden alle diagrammen weergeven en beschreven. Onder deze diagrammen vallen de ERD-, Use-Case- en Sitemap diagrammen. klassen diagram wordt later in dit document beschreven.

4.1 ERD



4.2 Use-Case

Dit is Use-Case van het systeem. Hierin staat hoe het programma te werk gaat.



- Een niet lid moet zijn strippenkaart kunnen inzien.
- Een niet lid moet zijn strippenkaart kunnen aanvullen.
- Een beheerder moet een aantal strippen van een strippenkaart kunnen strippen.
- Een beheerder moet een niet lid's strippenkaart kunnen inzien.
- Een beheerder moet een QR-code kunnen scannen van een niet lid.
- Een penningmeester moet betaalverzoek kunnen versturen.
- Een penningmeester moet betaalverzoeken kunnen inzien.
- Een penningmeester moet betalingen kunnen accepteren.
- Een penningmeester moet rechten kunnen aanpassen van een gebruiker.

5 User Interface

In dit hoofdstuk staan alle schetsen van de Android applicatie en web applicatie

5.1 Android

Dit zijn alle schetsen van de Android applicatie.

5.1.1 Menu's

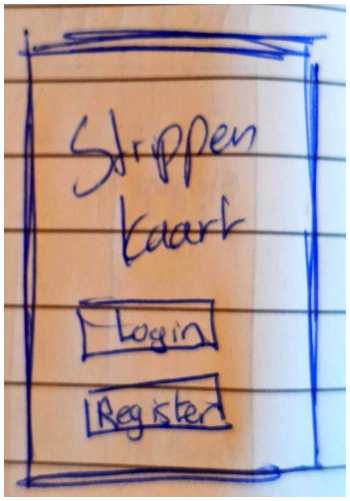
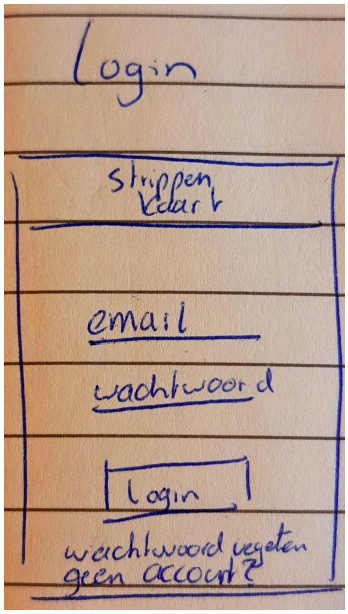
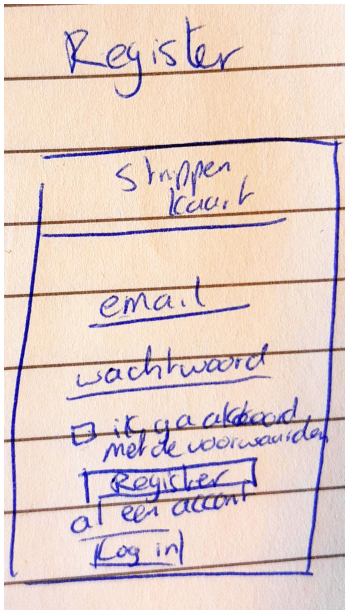


Figure 1: Dit is het startscherm. In dit scherm krijgt de gebruiker de keuze om in te loggen of als een nieuwe gebruiker te registreren

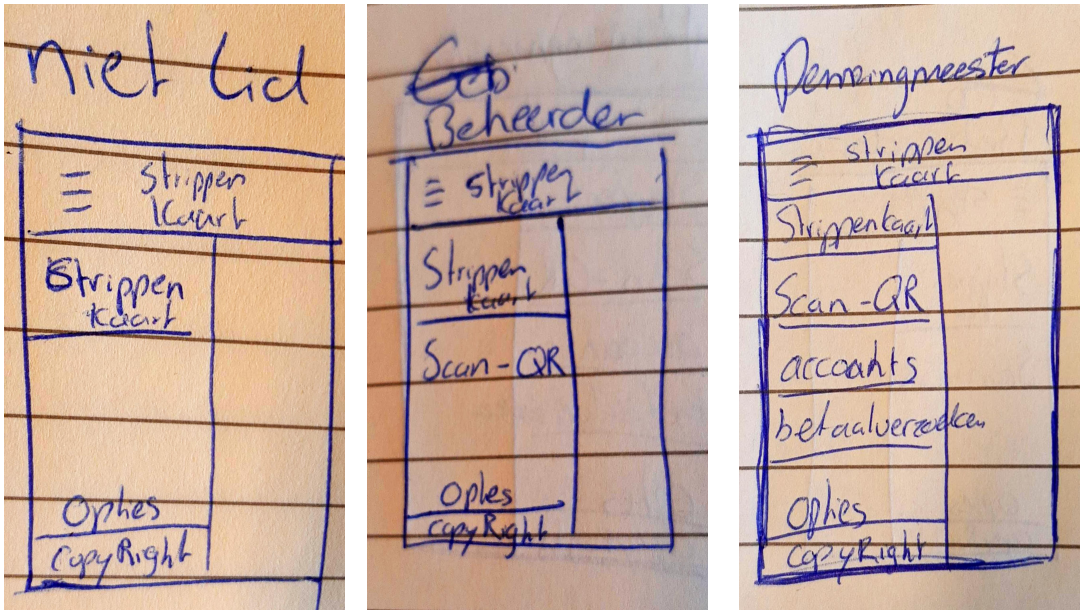


(a) Login scherm



(b) Register scherm

Figure 2: De Login en Registratie schermen. Hierin kunnen gebruikers inloggen en registreren. op elk scherm kunnen zij ook wisselen naar het ander scherm. In het login scherm kan de gebruik ook zijn wachtwoord weer veranderen als deze is vergeten.



(a) Menu niet lid (b) Menu beheerder (c) Menu penningmeester

Figure 3: Dit zijn de menu's voor de drie rollen. **a)** Het niet lid ziet alleen zijn strippenkaart. **b)** Een beheerder kan QR-codes scannen om strippen te strippen van een niet lid. **c)** Een penningmeester kan naar de accounts scherm en naar de betaalverzoeken scherm. Onderin het menu zit de opties.

5.1.2 Layout

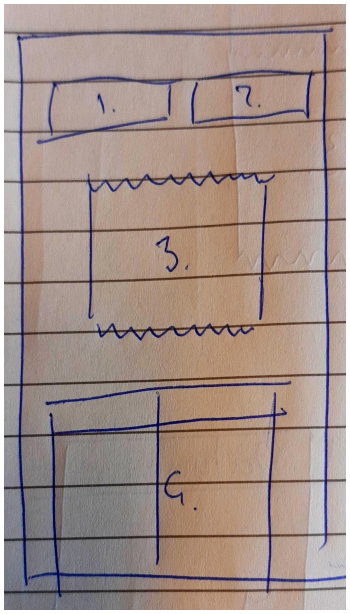


Figure 4: Scherm strippenkaart. **1)** Weergeef QR-code. **2)** Vul strippen aan. **3)** Aantal strippen over. **4)** geschiedenis van gebruikte en toegevoegde strippen.

5.2 Web

Dit zijn alle schetsen van de web applicatie.

5.2.1 Menu's

5.2.2 Layout

6 API Endpoints

In dit hoofdstuk staan alle endpoints beschreven van de API. Alle routes hebben een prefix van '/api/v1', bijvoorbeeld de login route: '/api/v1/login'.

6.1 Security

Dit zijn alle routes van de beveiling.

Method	Route	Parameters	Response	Beschrijving
POST	/login	Header: HTTP Basic	Access Token	Login in het systeem
POST	/register	Header: Bearer Token	N.v.t.	Registreer nieuw account

6.2 User

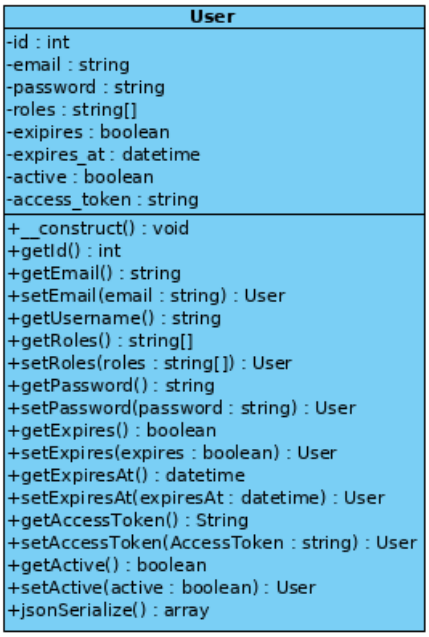
Dit zijn alle routes van de User Entity.

Method	Route	Parameters	Response	Beschrijving
GET	/user	N.v.t.	Alle users in DB	Haal alle account op
POST	/user/{id}	id: id van User Entity	Één User Entity	haal een account op
PATCH	/user/{id}	id: id van User Entity	N.v.t.	update een account
PATCH	/user/active/{id}	id: id van User Entity	N.v.t.	zet account op inactief of niet
PATCH	/user/roles/{id}	id: id van User Entity	N.v.t.	Verander rechten van een account
DELETE	/user/{id}	id: id van User Entity	N.v.t.	Verwijderd account uit DB.

7 Class Diagram

In dit hoofdstuk wordt de class diagrammen beschreven.

7.1 User



Property/Method	**Omschrijving**
id	De id van het account.
email	Het email adres van het account.
password	Het wachtwoord van het account.
roles	De rollen van het account.
expires	Als de access_token verloopt.
expires_at	De datum en tijd wanneer de access_token verlopen is.
active	Geeft aan als het account actief is.
access_token	De beveilings token die bij elk request gecontroleerd wordt.
__construct()	De constructor van het object.
getId()	De methode die het id ophaalt.
getEmail()	De methode die het email adres ophaalt.
setEmail(email)	De methode die het email adres zet in het object.
getUsername()	De methode die het email adres ophaalt.
getRoles()	De methode die de rollen ophaalt.
setRoles(roles)	De methode die de rollen zet in het object.
getPassword()	De methode die het wachtwoord ophaalt.
setPassword(password)	De methode die het wachtwoord zet in het object.
getExpires()	De methode de die expires ophaalt.
setExpires(expires)	De methode die expires zet in het object.
getExpiresAt()	De methode die expiresAt ophaalt.
setExpiresAt(expiresAt)	De methode die expiresAt zet in het object.
getAccessToken()	De methode die de accessToken ophaalt.
setAccessToken(accessToken)	De methode die de accessToken zet in het object.
getActive	De methode die active ophaalt.
setActive	De methode die active zet in het object.
jsonSerialize	De methode die het mogelijk maakt om om het object terug te sturen.

7.2 MultiTicket

7.3 PaymentRequest

8 Eisen

8.1 Systeemeisen

- Het syteem bestaat uit een API.
- Het basis-systeem is geschreven in Symfony 5.
- De mobile app is schreven voor Android.
- Het systeem maakt het mogelijk voor niet leden om met activiteiten van een vereniging mee te doen door te betalen met een strippenkaart.
- Het systeem bestaat uit:
  - Niet Leden
  - Beheerders
  - Strippenkaarten
  - Penningmeesters
  - Betalingen
  - Verengingen
- Strippenkaart moet worden verbonden met Gebruiker.
- De rechten van een gebruiker moet aangepast kunnen worden.
- De gebruiker moet strippenkaarten kunnen aanschaffen.
- De gebruiker moet inzicht hebben op zijn strippenkaarten.
- De penningmeester moet inzicht hebben op betalingen.
- De beheerder moet strippen van een strippenkaart kunnen strippen.
- De niet leden moeten QR-codes kunnen genereren.
- De beheerder moet QR-codes kunnen scannen.
- De app moet gebruik maken van het login systeem van Symfony.
- De penningmeester moet Tikkies kunnen sturen door het systeem.

8.2 Randvoorwaarden

8.2.1 Software/Systemen

Software	Versie
Symfony	5.0.5
MySQL	10.3-MariaDB
PHP	7.3
Android API	Level 24

9 Testplan

9.1 API

Unit testen Er wordt ge-unittest met CircleCI.

- Op alle routes worden gecheckt op de goede http response codes.
- Op alle routes worden gecheckt op de json response.
- Op alle routes worden laad tijden gecheckt.

9.2 Android

er wordt getest met:

- Unit-testen.
- Een gebruikersacceptatietest.
- Een functionele acceptatietest.

9.3 Web

er wordt getest met:

- Unit-testen.
- Een gebruikersacceptatietest.
- Een functionele acceptatietest.

10 Conventies/Styleguide

In dit hoofdstuk worden alle conventies behandeld. Onder deze conventies vallen Git en programeertalen.

Globale Conventies

- Maximaal één class per bestand.
- De class naam moet gelijk zijn aan de bestandsnaam.
- Er wordt gebruik gemaakt van spaties. Eén tab staat gelijk aan vier spaties.
- Alles wat in de code geschreven wordt, zowel variabelen als documentatie, moet geschreven worden in het *Engels*.
- Het gebruik van shortcode is toegestaan, hier wordt voorkeur aan gegeven.

10.1 Git

10.1.1 Branches

- Elke feature wordt in zijn eigen branche gemaakt.
- Elke feature branche stamt af van de development branch.
- Elke branch begint met een prefix. 'BUG' voor een bugfix, 'FEAT' voor een feature branch.
- Er wordt nooit direct gepushed naar de master branch of development branch.
- Er is altijd maar één persoon bezig met een branch. Als de branch aan een ander persoon over wordt gedragen, commit de vorige persoon niet meer in deze branch.

10.1.2 Commits

- Elke commit begint met de naam van de developer die gecommit heeft.
- Commits moeten klein blijven.

10.2 PHP

- Klasse namen moeten geschreven worden met UpperCamelCase.
- Functie/methode namen moet geschreven worden met camelCase.
- Properties en variabelen moet geschreven worden met camelcase.
- Alle documentatie wordt geschreven met PHPDoc.

Hiernaast worden de coding conventies van Symfony gevolgd. Deze conventies zijn **hier** te vinden.

10.3 Java

10.3.1 Linebreaks

- Regels niet langer maken dan 80-100 karakters (met uitzondering import regels).
- Er mag een linebreak komen na een komma en voor een operator.

10.3.2 Naming

- Klasse namen moeten geschreven worden met UpperCamelCase.
- Constant namen moeten geschreven worden in hoofdletters.
- Functie/methode/property/variable namen moet geschreven worden met camelCase.

10.3.3 Methodes Declaratie

Methodes worden op de volgende manier geschreven worden.

```
public void methodName(String firstParameterName, view viewParameterName) {
    //Method contents...
}
```

10.3.4 Statement Declaratie

Statements moet geschreven worden met een spatie voor en na de conditie.

```
if (condition) {
    //statement hier...
}
```

(a) Zo wel

```
if(condition){
    //statement hier...
}
```

(b) Zo niet

10.3.5 Return Statesments

- Een methode moet zo snel mogelijk ge-returned worden.
- Wanneer een methode een boolean returned aan de hand van één conditie, dan moet dat gedaan worden in een regel.

10.4 Javascript

- Alle functie en variable namen worden geschreven in camelCase.
- Alle name beginnen met een letter.
- Rond alle operators worden spaties geplaatst.
- Er worden 4 spaties gebruikt voor indentatie.
- Alle statements worden geëindigd met een ';'.

De rest van de guide is hetzelfde als bij PHP.

11 Plan van aanpak

11.1 Gemaakte afspraken

- Elke week wordt de opdrachtgever (Docent) op de hoogte gebracht van de vorderingen van het project.
- Alle code wordt geplaatst op GitHub. Alle document en releases worden op OneDrive gezet.
- Scrum wordt dagelijks bijgehouden in Trello.

11.2 Planning

Begin datum	Eind datum	Onderdeel
28/02/2020	06/03/2020	Documentatie
09/03/2020	20/03/2020	API
23/03/2020	17/04/2020	Android app

11.3 Akkoordverklaring

Indien beide partijen (opdrachtnemer(s) en opdrachtgever(s)) akkoort zijn met de analyse, worden zij geacht deze te ondertekenen ter bevestiging.

Collin Franckena, student & opdrachtnemer	Datum
Jan Zuur, docent & opdrachtgever	Datum