Project IntervisieSpel

Kwaliteitseisen document

Inhoud

[1.0 Teamleden & Rollen 1](#_Toc51056741)

[2.0 Applicatie 1](#_Toc51056742)

[2.1 Doel 1](#_Toc51056743)

[2.2 Aard 1](#_Toc51056744)

[2.3 Doelgroepen 1](#_Toc51056745)

[3.0 Agile/Scrum 2](#_Toc51056746)

[3.1 Verantwoordelijkheden 2](#_Toc51056747)

[3.2 Definition of Done 2](#_Toc51056748)

[4.0 Architectuur en Softwarepatronen 2](#_Toc51056749)

[4.1 MVC 2](#_Toc51056750)

[4.1.1 Voordelen 2](#_Toc51056751)

[4.1.2 Nadelen 3](#_Toc51056752)

[4.2 SPA 3](#_Toc51056753)

[4.2.1 Voordelen 3](#_Toc51056754)

[4.2.2 Nadelen 3](#_Toc51056755)

[4.3 MPA 3](#_Toc51056756)

[4.3.1 Voordelen 3](#_Toc51056757)

[4.3.2 Nadelen 3](#_Toc51056758)

[4.4 Keuze architectuur en softwarepatronen: 3](#_Toc51056759)

[5.0 Frameworks en Libraries 4](#_Toc51056760)

[5.1 Laravel 4](#_Toc51056761)

[5.1.1 Voordelen 4](#_Toc51056762)

[5.1.2 Nadelen 4](#_Toc51056763)

[5.2 Lumen 4](#_Toc51056764)

[5.2.1 Voordelen 4](#_Toc51056765)

[5.2.2 Nadelen 4](#_Toc51056766)

[5.3 Angular 5](#_Toc51056767)

[5.3.1 Voordelen 5](#_Toc51056768)

[5.3.2 Nadelen 5](#_Toc51056769)

[5.4 Vue 5](#_Toc51056770)

[5.4.1 Voordelen 5](#_Toc51056771)

[5.5 Keuze frameworks en libraries 5](#_Toc51056772)

[5.5.1 Vue 6](#_Toc51056773)

[5.6 Hoofcomponenten 6](#_Toc51056774)

[5.7 Systeemomgeving 6](#_Toc51056775)

[5.8 Onderlinge samenwerking tussen hoofdcomponenten en systeemomgeving 6](#_Toc51056776)

[6.0 Database 7](#_Toc51056777)

[6.1 Logische datastructuur 7](#_Toc51056778)

[6.2 Gekozen implementatie en onderbouwing 7](#_Toc51056779)

[7.0 Coding standards 8](#_Toc51056780)

[7.1 Branching & commits 8](#_Toc51056781)

[7.1.1 Branches 8](#_Toc51056782)

[7.1.2 Commits 8](#_Toc51056783)

[7.2 Overige standards 9](#_Toc51056784)

[7.2.1 Controller/model conventies 9](#_Toc51056785)

[7.2.2 Component(vue) conventies 9](#_Toc51056786)

[7.2.3 SASS conventies 9](#_Toc51056787)

[7.2.4 Navigatie conventies 9](#_Toc51056788)

[7.2.5 Overige standards 9](#_Toc51056789)

[8.0 Eisen user interface 10](#_Toc51056790)

[8.1 Look & Feel 10](#_Toc51056791)

[8.2 Navigatie 10](#_Toc51056792)

[8.3 Invoer 10](#_Toc51056793)

[9.0 Organisatie 10](#_Toc51056794)

[9.1 Rolverdeling 10](#_Toc51056795)

[9.2 Bereikbaarheid 11](#_Toc51056796)

[9.3 Werkafspraken 11](#_Toc51056797)

# 1.0 Teamleden & Rollen

|  |  |
| --- | --- |
| Naam | Rol |
| Dhr. de Ruijter | Product Owner |
| Mevr. Schlundt Bodien | Product Owner |
| Jesse Sommeling | Scrum Master, Versiebeheerder, Developer |
| Danny van der Weijde | Databasebeheerder, Developer |

# 2.0 Applicatie

## 2.1 Doel

**DL-01** Het doel van de applicatie is dat je het interventie spel online kunt spelen.

**DL-02** In de applicatie zijn de volgende informatievoorzieningen belangrijk:

* 1. Gebruikersinformatie (naam, email, password, etc.)
  2. Kaartgegevens (categorie, vragen, etc.)
  3. Regels van intervisiespel

**DL-03** In de applicatie komen (mogelijk) de volgende transacties aan bod:

1. Bestelling van licentie.

## 2.2 Aard

**AA-01:** De applicatie is een intervisie spel om aanvaringen of problemen mee op te lossen op de werkvloer.

**AA-02:** Er zijn nog geen andere applicaties m.b.t. intervisie zo ver bekend.

**AA-03:** Een webapplicatie dat hoogstwaarschijnlijk op deze applicatie zal lijken, is Kahoot. Hierbij kan men denken aan het kunnen aansluiten bij een groep op basis van een pincode o.i.d. om een spel te spelen.

## 2.3 Doelgroepen

Het spel doet het al heel goed in de fysieke vorm, maar nu dat je veel thuis moet blijven of van thuis moet werken is de vraag naar dit spel groot geboren om online ook toegankelijk te zijn.

Het spel wordt voornamelijk gebruikt voor bedrijfsproblemen en aanvaringen. Maar dit kan ook gebruikt worden om juist de band tussen collega’s te verbeteren.

# Agile/Scrum

## 3.1 Verantwoordelijkheden

**VER-01:** De **Scrum Master** is verantwoordelijk voor het op correcte wijze uitvoeren van de Scrum aanpak. Dit wordt gedaan door ervoor te zorgen dat het hele team de Scrum theorie, werkwijzen en regels naleeft. De Scrum Master communiceert met de Product Owner. Tevens heeft de Scrum Master op het moment ook de leiding over het versiebeheer.

**VER-02: Developers** nemen op eigen wijze de ontwikkeling van de applicatie in handen.

## 3.2 Definition of Done

Doorgaans is de meest gebruikte toepassing van de Definition of Done (hierna te noemen DoD) dat de Product Owner (JDR) het product beoordeeld heeft en de user story geaccepteerd heeft. De userstory is pas voltooid indien het aan de volgende criteria heeft voldaan:

**DoD-1: Unit tests zijn geslaagd (indien van toepassing).**

**DoD-2: Code is beoordeeld door andere programmeur(s) op functionaliteit.** (Alle must-have’s zijn geïmplementeerd.)

**DoD-3: Er wordt gekeken naar of de code is geschreven naar de regels die de groep er over gemaakt heeft.**

**DoD-4: De code is gedocumenteerd en voldoet aan de afgesproken layout/coding standaarden.**

**DoD-5: De userstory/feature is getest door minimaal één ander persoon.**

* + Teststappen worden beschreven
  + Test wordt uitgevoerd
  + Resultaat van test is beschreven

**DoD-6: De code is gecommit via een pullrequest in het versiebeheersysteem met een verwijzing naar de userstory/feature.**

# 4.0 Architectuur en Softwarepatronen

## 4.1 MVC

MVC staat voor **Model-View-Controller**. Het is een architectuur of een softwareontwerppatroon dat het maken van enorme applicaties eenvoudig maakt. Het behoort niet tot een specifieke programmeertaal of raamwerk, maar het is een concept dat je kunt gebruiken bij het maken van elke soort applicatie of software in elke programmeertaal.

Als u bijvoorbeeld een applicatie in PHP ontwikkelt, kun je frameworks zoals Laravel of Codeigniter gebruiken die MVC-architectuur gebruiken om je te helpen applicaties snel en eenvoudig te ontwikkelen.

### 4.1.1 Voordelen

1. De ontwikkeling van de applicatie gaat snel.
2. Makkelijk voor meerdere ontwikkelaars om samen te werken.
3. Makkelijker om de applicatie bij te werken.
4. Gemakkelijker te debuggen omdat er meerdere niveaus zijn binnen de applicatie.

### 4.1.2 Nadelen

1. In het begin is het vrij moeilijk om de MVC structuur te begrijpen.
2. Niet geschikt voor kleinere applicaties omdat het negatieve effecten kan hebben in de prestaties van de applicatie en het ontwerp ervan.

## 4.2 SPA

Ook wel een Single Page Application genoemd, is een [webapplicatie](https://nl.wikipedia.org/wiki/Webapplicatie) of [website](https://nl.wikipedia.org/wiki/Website) die past op een enkele [webpagina](https://nl.wikipedia.org/wiki/Webpagina) met het doel een vlotte gebruikerservaring te bieden. In een SPA wordt alle benodigde code - [HTML](https://nl.wikipedia.org/wiki/HyperText_Markup_Language), [JavaScript](https://nl.wikipedia.org/wiki/JavaScript) en [CSS](https://nl.wikipedia.org/wiki/Cascading_Style_Sheets) - opgehaald met een enkele laadactie van de pagina, of de noodzakelijke middelen worden dynamisch geladen en aan de pagina toegevoegd zodra ze nodig zijn, meestal als reactie op acties van de gebruiker. De pagina wordt op geen enkel moment in het proces opnieuw geladen.

### 4.2.1 Voordelen

1. Reactief (illusie geen pagina laden.)
2. Goede laadsnelheden
3. Goede user experience
4. Front-end en back-end zijn goed van elkaar gescheiden

### 4.2.2 Nadelen

1. SEO blijft een uitdaging.
2. Beveiliging wordt bemoeilijkt (end points beveiligen i.p.v. individuele pagina’s).
3. Oudere browsers kunnen SPA’s vaak minder goed draaien.

## 4.3 MPA

Ook wel een **Multi Page Application** genoemd. Deze maakt gebruik meerdere pagina’s om de front-end met de back-end te laten communiceren. Voor elk request worden de HTML, CSS en scripts van een pagina opnieuw opgehaald en ingeladen.

### 4.3.1 Voordelen

1. Goede SEO mogelijkheden
2. Eenvoudiger te doorgronden door natuurlijke structuur van pagina’s.

### 4.3.2 Nadelen

1. Grote applicaties kunnen trager werken door het constant compleet herladen van pagina’s.
2. Front-end en back-end vaak ‘tightly coupled’

## 4.4 Keuze architectuur en softwarepatronen:

Er is gekozen voor een Single Page Application (hierna te noemen SPA).

Een SPA is erg reactief. In die zin zou je kunnen zeggen dat het lijkt op een mobiele of desktop app. Het reageert gelijk als je op iets klikt en het geeft je de illusie dat het geen enkele pagina opnieuw hoeft te laden. Laadsnelheden en user experience zijn erg goed. Een MPA daarentegen kan trager werken, door het continu opnieuw laden van de pagina. Dit was een primaire afweging waarom er gekozen is voor een SPA structuur.

Front-end en back-end zijn goed van elkaar gescheiden. Je hoeft alleen een API te creëren die informatie vanuit de database stuurt naar de front-end. Styling, structuur en logica gebeurd (voornamelijk) op de front-end. In een MPA is de front-end en back-end vaak ‘tightly coupled’ waardoor kleine veranderingen in een applicatie vaak bemoeilijkt wordt des te groter de applicatie wordt.

**SEO blijft echter een uitdaging;** Search Engine Optimization is moeilijker met een SPA dan een MPA programma. Bij websites die JavaScript-based (zoals SPAs) zijn wordt het indexatie proces bemoeilijkt. Om de informatie op te slaan is er een extra stap benodigd van Googlebot. Alle JavaScript moet worden uitgevoerd of ‘gerenderd’ om je content en links te kunnen indexeren.

# 5.0 Frameworks en Libraries

## 5.1 Laravel

Laravel framework is erg populair voor softwareontwikkeling op maat. Het is het PHP-framework met de meeste sterren op Github: meer dan 56.000 ontwikkelaars van over de hele wereld (voornamelijk uit de VS) waarderen de robuuste functies van dit platform enorm. Waarom is Laravel zo populair? Het biedt de juiste tools om websites en webapps sneller, stabieler en zeer gemakkelijk te onderhouden te bouwen.

### 5.1.1 Voordelen

1. Gemakkelijke integratie van autorisatie.
2. Veel libraries tot zijn beschikking.
3. Veiligheid hoog in het vaandel (SQL injection, cross site requests en cross site scripting).
4. Leesbare, overzichtelijke code.
5. Gemakkelijk uitbreidbaar.

### 5.1.2 Nadelen

1. Werkt niet goed met shared hosting (er zijn omleidingen voor nodig om het te laten werken).

## 5.2 Lumen

Lumen is in 2015 gemaakt door Taylor Otwell, die ook de maker is van het Laravel-framework. Het is een open-source, ultralicht framework dat voornamelijk is ontworpen voor het schrijven van microservices in PHP. Lumen is een micro-framework, het betekent dat het een afgeslankte, snelle versie is van een volwaardig framework voor het schrijven van webapplicaties, in dit geval is het de jongere broer van het Laravel-framework.

### 5.2.1 Voordelen

1. Zeer snel.
2. Eenvoudige syntaxis.
3. Zeer gemakkelijke migratie van het project naar het volwaardige Laravel-framework.
4. Lumen biedt een snelle routefunctie dankzij de Fast Route-bibliotheek. Deze bibliotheek biedt implementatie van routing op basis van Regular Expressions.

### 5.2.2 Nadelen

1. Helaas ondersteunt Lumen geen sessions, maar biedt het een eenvoudig authenticatiemechanisme op basis van tokens.

## 5.3 Angular

Angular is een typescript gebasseerde framework en development platform voor het maken van efficiënte en ingewikkelde single page applications. Het is opensource en heeft nauwe banden met Google.

### 5.3.1 Voordelen

1. Het wordt ondersteund door Google.
2. Heeft een brede selectie aan libraries.
3. Snelle laadtijden en goede beveiliging

### 5.3.2 Nadelen

1. SEO wordt bemoeilijkt.
2. Flinke leercurve

## 5.4 Vue

VueJS is een open source JavaScript framework. Het wordt gebruikt voor het maken van efficiënte en ingewikkelde single page applications. Het is open source en een rijzende ‘ster’ binnen de development community.

### 5.4.1 Voordelen

1. Gemakkelijk te leren
2. Gedetailleerde documentatie
3. Snelle laadtijden (kleine framework om te laden.)
4. Simpele integratie

## 5.5 Keuze frameworks en libraries

Er is gekozen voor Laravel met een SPA structuur met behulp van Vue.

Naast dat de klant wenst om een applicatie te hebben dat relatief snel tot stand wordt gebracht en voor ons als ontwikkelaars (met ervaring met Laravel) sneller om te werken is dan met andere frameworks , zijn er nog andere redenen:

Laravel staat bekend om zijn mooie, simpele en leesbare code. Hierdoor zijn Laravel applicaties vaak stabiel, betrouwbaar en goed te onderhouden en uit te breiden.

Binnen een paar minuten kun je taken die -normaal gesproken- veel tijd kosten binnen een paar minuten bouwen. Dit scheelt ontzettend veel tijd bij het onderhouden van de software en in het ontwikkelproces.

Er is gedetailleerde documentatie aanwezig voor developers. Het is open source, dus meer dan 30.000 andere developers tot je beschikking die kunnen helpen in plaats van dat je afhankelijk bent van één partij voor de informatie.

Veel sites en applicaties zijn afhankelijk van andere tools of software. Met Laravel is het gemakkelijk om die tools en software te integreren.

Laravel heeft veiligheid hoog in het vaandel staan. Standaard biedt Laravel goede bescherming tegen SQL injection, cross site requests en cross site scripting. Gezien de klant snel een werkende applicatie zou willen, moet deze ook gegarandeerd veilig zijn. Ook gezien er mogelijk in de toekomst gewerkt gaat worden met betaalde licenties.

Vaak moet authenticatie van gebruikers zelf geschreven worden. Dit is een tijdrovend proces. Integratie van authenticatie is gemakkelijker gemaakt met Laravel. Hierdoor kan men, binnen enkele minuten, een gehele authenticatie opzetten.

### 5.5.1 Vue

Gezien Laravel bekend staat om zijn leesbare code, stabiliteit en betrouwbaarheid is er gekozen om er Vue bovenop te gebruiken als javascript-library (gezien er gekozen is voor een SPA structuur). Vue heeft als voordeel boven Angular en React is dat het gemakkelijker te leren is, overzichtelijker is en dus gebruiksvriendelijker is.

Gemakkelijker te leren omdat html, css en javascript in een soortgelijke structuur blijft als men dat zonder javascript library zou toepassen. Overzichtelijker omdat het uit zichzelf styling (css) en logica (javascript) per component overzichtelijk kan scheiden. Bovendien is Vue goed te combineren met Laravel. Laravel zorgt voor de back-end, terwijl Vue de front-end voor zijn rekening neemt.

## 5.6 Hoofcomponenten

De hoofdcomponenten zijn:

**HC-01:** Client

**HC-02:** Server

## 5.7 Systeemomgeving

**SO-01:** Laravel 7/8

**SO-02:** Laravel 7/8 API

**SO-03:** Laravel Sanctum/Jetstream

**SO-04:** MySQL database

**SO-05:** VueJs

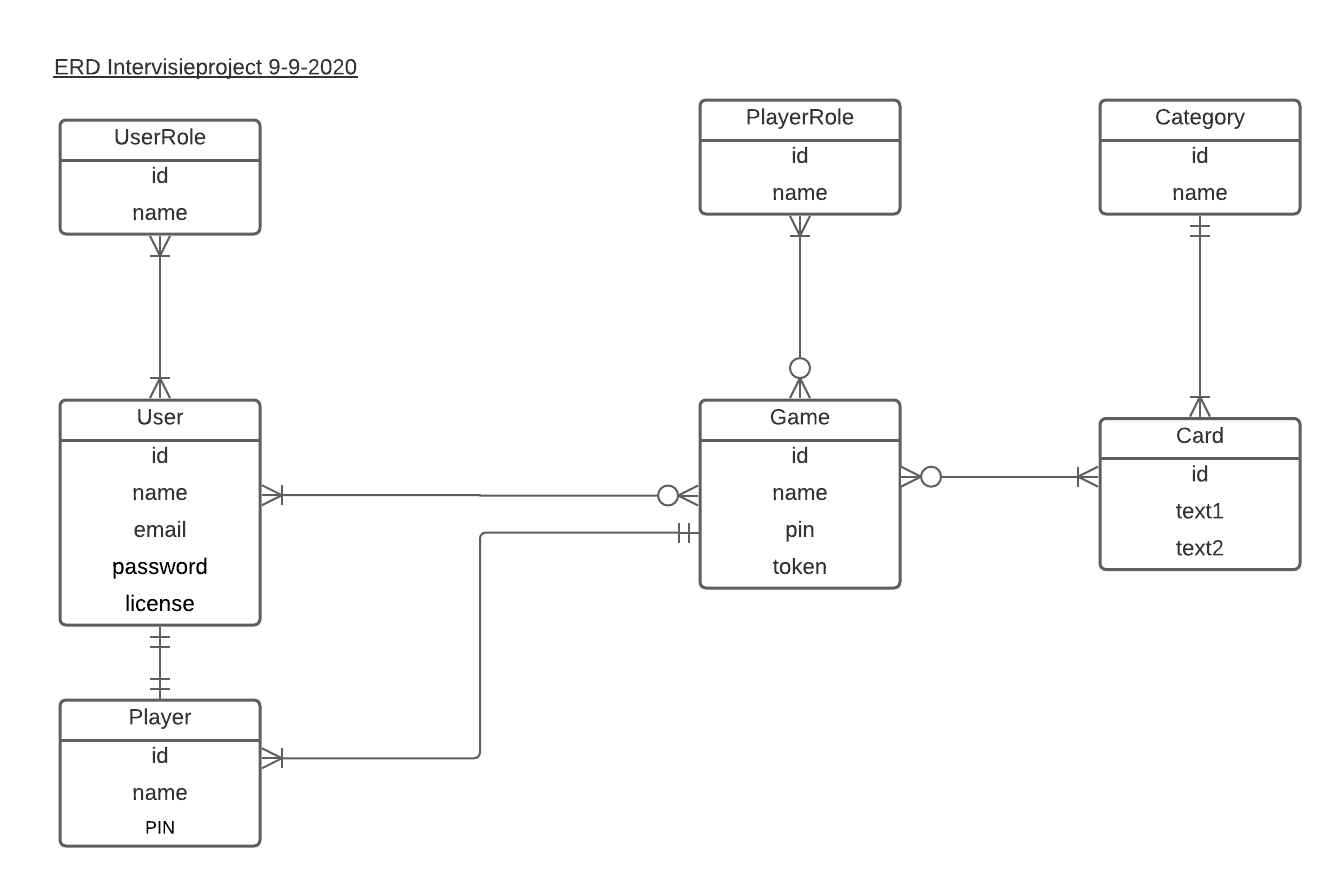
**SO-06:** Pusher/ Laravel Echo

## 5.8 Onderlinge samenwerking tussen hoofdcomponenten en systeemomgeving

Op de server staat er een Laravel applicatie, wat door de client wordt opgevraagd. Eenmaal opgevraagd communiceert de client met de database dat tevens op de server staat, indien nodig. Vue maakt via Axios (library voor JavaScript promise-objecten) een call aan naar de Laravel API om informatie op te vragen. Daarnaast wordt er gebruik gemaakt van Laravel Sanctum voor de authenticatie. Tot slot wordt er gebruik gemaakt van Pusher om realtime updates te verkrijgen middels websockets van de verkregen informatie uit de database.

# 6.0 Database

## 6.1 Logische datastructuur



## 6.2 Gekozen implementatie en onderbouwing

NoSQL krijgt snel databasevervuiling omdat relaties moeilijker beschreven/aangegeven kunnen worden dan een relationele database en er dus vaak gegevens dubbel ingezet moeten worden. NoSQL heeft wel weer als voordeel dat, alhoewel het niet echt gebruik maakt van relaties (dit is wel mogelijk, maar onhandiger dan bij een relationele database) is het wel mogelijk is om binnen één call heel veel informatie op te roepen. Tevens werkt het ook sneller dan een relationele database.

Echter, gezien het hier niet gaat om duizenden records die tegelijk opgehaald behoeven te worden en de database zichzelf ‘schoon’ houdt, valt de keuze op een relationele database. MySQL leent zich hier prima voor. Laravel ondersteund dit en heeft een integratie van MySQL.

# 7.0 Coding standards

Op 4-9-20 zijn er standaarden afgesproken m.b.t. het schrijven van code. Deze paragraaf gaat over de vastgelegde regels.

**CS-01: Code wordt gedocumenteerd m.b.v. blok commentaar.** Zo kun je denken aan blok commentaar voor het beschrijven van functions en/of methods.

**CS-02: Inline commentaar wordt alléén gebruikt ter verheldering van geschreven code.**

**CS-03: Variabele- en constantennamen zijn altijd in het Engels**

**CS-04: Gebruik van juiste tab-afstanden is vereist.** Bij iedere element/function/etc. is een tab inspringen verplicht.

**CS-05: Constanten altijd in UPPERCASE**

**CS-06: Classenamen in PascalCase**

**CS-07: Variabele en function namen in camelCase.**

**CS-08: Logisch gebruik van variabelen en constanten.**

**CS-09: Gebruik van DRY (Don’t repeat yourself) principe.**

**CS-10: Gebruik van Single Responsibility principe op o.a. functions.**

**CS-11: Code moet herbruikbaar zijn. (code reusability)**

**CS-12: Code wordt zo simpel en klein mogelijk gehouden.** Regelmatig refactoren a.d.v. feedback code reviews en eigen input na schrijven van code.

**CS-13: Unittests schrijven voor methods wanneer er extra classes aangemaakt zijn.**

**CS-14: Objectnamen altijd in camelCase.**

## 7.1 Branching & commits

### 7.1.1 Branches

Er zijn diverse branches waar op gecommit kan worden.

**BR-01:** De **master branch** waar (voorlopige) releases op komen.

**BR-02:** De **developers branch** waar voltooide features samengevoegd worden en getest worden door andere developers.

**BR-03: Individuele branches** waar features worden gebouwd voordat deze op de developers branch worden samengevoegd.

### 7.1.2 Commits

**CO-01: Alle** commits zijn voorzien van een volledige beschrijving.

**CO-02:** Alle commits zijn voorzien van een featurecode te vinden in de trello-board.

**CO-03:** Per commit wordt er verwezen naar de desbetreffende feature van de trello-board

## 7.2 Overige standards

### 7.2.1 Controller/model conventies

**CM-01:** Controllers worden in enkelvoud met hoofdletter beschreven + ‘Controller’ als postfix.

Bijv. GameController

**CM-02:** Controllers die bij een bepaalde subgroep behoren komen in een aparte map.

Bijv. Api controllers behoren in een mapje genaamd Api.

**CM-03:** Models worden in enkelvoud beschreven met hoofdletter

Bijv. Game

### 7.2.2 Component(vue) conventies

**CC-01:** Alle Vue components behoren in een map per subcategorie.

**CC-02:** Bij alle Vue components worden de titels beschreven in PascalCase.

**CC-03:** Imports/registraties van components zijn altijd PascalCase.

### 7.2.3 SASS conventies

SASS is een manier om CSS overzichtelijk en opgedeeld te beschrijven.

**SA-01:** Het hoofdcomponent blijft genaamd ‘app.scss’ en blijft direct in de sass-map.

**SA-02:** De submappen voor de sass-map zijn:

1. abstracts (met mixins, variables, functions etc.)
2. base (met animations, base, typography, etc.)
3. components (met buttons etc.)
4. layout (met header, footer etc.)
5. pages(overige scss/css voor pagina’s)

### 7.2.4 Navigatie conventies

**NC-01:** Alle URL’s volgen de volgende opbouw:

1. controller/action
2. controller/controllerparameter/action
3. controller/action/actionparameter
4. controller/controllerparameter/action/actionparameter

**NC-02:** Indien er sprake is van een afscheiding. Dan altijd de naam van de afscheiding als prefix. Bijvoorbeeld een route in het admin gedeelte ziet er dan als volgt uit:

1. admin/controller/action

### 7.2.5 Overige standards

**OS-01:** Iedere html pagina moet responsive benaderd kunnen worden. Dat wil zeggen: een uiterlijk voor grote, middel en kleine beeldschermen.

# 8.0 Eisen user interface

## 8.1 Look & Feel

De klant wilt dat de app er rustig en vertrouwd uitziet. Dit betekent dat ze dezelfde kleuren wilt gebruiken als die voor de kaarten en als de website van Cinop zelf. Voor stijlvoorbeelden zie de website van Cinop en het kaartspel. Volgens de klant is er van tevoren nagedacht over de kleuren die op de kaarten staan, en die wilt ze ook behouden.

## 8.2 Navigatie

Er moet een vast menu constant zichtbaar zijn, tot je het spel in gaat. Dit betekent dat er verschillende menuopties aanwezig zijn op de (horizontale) navigatiebalk. De navigatiebalk bevat een ‘home-‘, ‘uitleg-‘ en een ‘kopen’ knop. Verder staat er een knop met ‘inloggen’ en ‘account aanmaken’.

Gezien de applicatie geen diepe menu’s bevat, komen er ook geen breadcrumbs. Betreffende de opbouw van de URL wordt er doorverwezen naar punt 7.2.4 van dit document.

## 8.3 Invoer

Bij de username invoer zijn er alleen alfabetische karakters toegestaan, met een aantal vooraf geselecteerde speciale characters (zoals de ‘ –‘ of de ‘ \_’). Voor de rest bij de invoer van tekst is het mogelijk om alfabetische karakters te gebruiken en speciale tekens zoals de vraagteken en punt.

Er wordt aangegeven dat invoer verplicht is middels een rode ster met een commentaar er bij indien de formulier niet volledig is ingevuld. Client-side betekent dit dus een ‘required’ tag in de html, server side betekent dit mede een controle op invoer als een rode ster met een rood berichtje terug op de client.

Als feedback krijgt men (client side): *Deze invoerveld is verplicht!* Server-side krijgt men feedback terug als: *Karakters zijn niet toegestaan.*

Middels tailwindCSS en Jquery/Javascript krijgt men een berichtje te zien op het scherm dat een transactie of statusverandering geslaagd is.

# 9.0 Organisatie

## 9.1 Rolverdeling

* **Scrum Master:** Jesse Sommeling
* **Databasebeheerder:** Danny van der Weijde
* **Versiebeheer:** Jesse Sommeling
* **Contactpersoon:** Danny van der Weijde
* **Notuleren:** Jesse Sommeling
* **Beheren testomgeving:** Danny van der Weijde

## 9.2 Bereikbaarheid

* Jesse Sommeling xxxxxxxx xxxxxxxx
* Danny van der Weijde xxxxxxxx xxxxxxxx

## 9.3 Werkafspraken

In principe zijn we altijd aanwezig bij de lessen. Buiten de lessen om is het een eigen keuze of je aanwezig/online bent of niet. Bij doktersafspraken e.d. op tijd (minimaal 24u) van tevoren doorgeven aan teamgenoten.

Bij afwezigheid kan er gecommuniceerd worden via Teams (groep).

Indien er onvoldoende inzet wordt getoond tijdens het project of de bovenstaande regels niet worden nageleefd dan volgt er een waarschuwing. Bij drie waarschuwingen wordt je van het project verwijderd.