Projectlab Bachelor EO-ICT

Quizcomputer

Francis Hoste

Gerd Van Buggenhout

Jonas Van Waes

Kevin Van Hee

Tomas Kello

Academiejaar 2014-2015

Inhoud

[2 Figurenlijst 3](#_Toc419924504)

[3 Productomschrijving (FH) 4](#_Toc419924505)

[4 Hardware beschrijving 4](#_Toc419924506)

[4.1 LED-Board (FH) 4](#_Toc419924507)

[4.2 Microcontroller (KVH) 5](#_Toc419924508)

[4.3 Voeding (KVH) 8](#_Toc419924509)

[4.4 Netwerkinfrastructuur (FH) 11](#_Toc419924510)

[5 Software beschrijving 11](#_Toc419924512)

[5.1 Database (FH) 11](#_Toc419924513)

[5.2 Front-end 13](#_Toc419924514)

[5.2.1 Zend Framework (JVW) 13](#_Toc419924515)

[5.2.2 MVC-model (JVW) 14](#_Toc419924516)

[5.2.3 Applicatie layout (JVW) 15](#_Toc419924517)

[5.2.4 Specifieke pagina's (JVW) 18](#_Toc419924518)

[5.2.5 De Quiz-pagina (GVB) 19](#_Toc419924519)

[5.2.6 Uploaden van afbeeldingen 23](#_Toc419924520)

[5.3 Serversoftware 25](#_Toc419924521)

[5.4 Stuurprogramma Microcontroller 25](#_Toc419924522)

[6 Toekomstig Werk 25](#_Toc419924523)

[6.1 Beveiliging 25](#_Toc419924524)

[6.2 Compatibiliteit verschillende browsers 25](#_Toc419924525)

[7 Conclusie 25](#_Toc419924526)

# Figurenlijst

Afbeelding 1 : 16x32 LED board 5

Afbeelding 2: Blokschema Mbed 7

Afbeelding 3: Mbed als microcontroller 8

Afbeelding 4: schema van de voeding 9

Afbeelding 5 : PCB design van de voeding 10

Afbeelding 6: netwerk architectuur 11

Afbeelding 7 : EER diagram 12

Afbeelding 8: MVC model 15

Afbeelding 9 : framework workflow 16

Afbeelding 10: Question Workflow 20

Afbeelding 11 : codefragment Timer 21

Afbeelding 12 : communicatie met Dbase 22

Afbeelding 13 : image upload 23

Afbeelding 14 : workflow Image upload 24

# Productomschrijving (FH)

We maken een computergestuurd examen dat centraal vanuit een server wordt gestuurd. Clients kunnen via een switch connecteren aan de server, zich aanmelden als admin, examinator en als deelnemer.

De afbeeldingen die bij de vragen horen worden op het scherm van de gebruiker getoond en synchroon op een LED-board.

In de database worden vragen opgesteld, afbeeldingen bijgehouden.

# Hardware beschrijving

## LED-Board (FH)

We hebben beslist om een LED-matrix bord aan te schaffen.

De relatief hoge kostprijs voor afzonderlijke RGB-LED’s en onze ambitie om minstens een 16x16 matrix te hebben, heeft ons in de richting van deze aankoop geleid.

Het is de bedoeling om minstens een leesbare afbeelding op de matrix te brengen. We vrezen dat met een resolutie kleiner dan 16x16 zelfs een eenvoudige afbeelding als een verkeersbord niet herkenbaar zal zijn.

Via de website van Adafruit.com vinden we een geschikt en betaalbaar model met volgende specs:

* 16x32 RGB LED matrix panel
* 5Vdc input en wanneer alle leds branden 2,5A
* Chainable (voor eventuele uitbreiding)

Downs:

Er is geen PWM driver voorzien. We gaan onze ARM M3 mbed gebruiken om de LED’s rechtstreeks aan te sturen.



Afbeelding : 16x32 LED board

## Microcontroller (KVH)

De microcontroller die we hebben gekozen voor dit project is de mbed LPC1768.

De reden hiervoor is dat we al reeds ervaring hebben met het programmeren van de mbed, met gevolg dat we hierdoor heel wat tijd uitsparen met het bestuderen van de specificaties van de microcontroller.

De mbed LPC1768 is ontwikkeld om verschillende systemen te gaan prototypen, waaronder vooral systemen met ethernet en USB. Daarnaast biedt de mbed ook flexibiliteit aan voor tal van perifere interfaces.

De microcontroller bevat ook een ingebouwde USB FLASH programmer, waardoor extra programmeerkits en dergelijke niet moeten aangeschaft worden.

Enkele noemenswaardige specificaties:

* 32-bit ARM Cortex-M3 core
* 512kB FLASH
* 32kB RAM
* Ethernet
* USB
* SPI
* I2C
* ADC
* DAC
* PWM
* Andere I/O interfaces

|  |
| --- |
| Afbeelding 2: Blokschema Mbed |
|  |

In Afbeelding 3 worden de interacties weergegeven waarbij de microcontroller centraal staat. Zo is te zien dat de potentiometer en de joystick worden gebruikt als inputs. Hierbij wordt de potentiometer gebruikt om de lichtintensiteit van de ledmatrix te besturen en de joystick als een reset voor de matrix.

De ledmatrix zelf wordt als output gebruikt van de mbed.

De communicatie tussen de mbed en de webserver gebeurd via het TCP/IP protocol over ethernet. Om de ethernet verbinding tot stand te brengen wordt gebruik gemaakt van de RJ45 ethernet connector die aanwezig is op de mbed application board.

|  |
| --- |
| Afbeelding 3: Mbed als microcontroller |
|  |

## Voeding (KVH)

De adafruit 16x32 LED-matrix vereist een voeding van 5V die in staat is een stroom te leveren tot 2A.

Hierdoor hebben we besloten om zelf een voeding te maken die aan deze specificaties voldoet.

|  |
| --- |
| Afbeelding 4: schema van de voeding |
|  |

De voeding (zie Afbeelding 4) krijgt een netspanning van ongeveer 230V wisselspanning aangelegd aan de ingang. Deze spanning wordt door de transformator omgezet naar een spanning van 9V. De bekomen 9V wisselspanning wordt dan met behulp van een bruggelijkrichter omgezet naar een gelijkspanning. De gelijkspanning van 9V wordt op zijn beurt door middel van een spanningsregelaar aangepast tot de gewenste spanning van 5V.

De schakeling bevat verder nog een LED als indicator dat de voeding is aangesloten aan de netspanning.

Van de componenten werd telkens onderzocht of deze in staat waren de opgelegde maximumstroom van 2A te kunnen leveren.

Na het vervolledigen van het schema werd overgegaan tot het PCB design van de voeding (zie Afbeelding 5).

|  |
| --- |
| Afbeelding 5 : PCB design van de voeding |
|  |

Het resultaat van dit ontwerp was een enkelzijde PCB.

Na het etsen van plaat werden alle componenten op de PCB gesoldeerd en kon deze getest worden.

Het testen gebeurde door de LED-matrix aan te sluiten op de voeding. Eerst werd de uitgangspanning gemeten aan de uitgang van de voeding. Deze spanning was de gewenste 5V. Daarna werd ook telkens de stroom gemeten wanneer de LEDs werden aangezet. Deze stroom naderde bij hoge belasting tot 2A. Door het plaatsten van enkele koelelementen werd de opwarming van de componenten wat tegengegaan. Het resultaat was dus zoals gewenst: een werkende voeding met uitgangsspanning van 5V die een stroom kon leveren van 2A.

## Netwerkinfrastructuur (FH)

Om ons systeem zo veel mogelijk multi-inzetbaar te maken en om de mogelijkheid te bieden om een zo groot mogelijk aantal gebruikers te verbinden met onze toepassing, hebben we gekozen om een client-server-architectuur op te zetten.

Hiervoor gebruiken we een switch waaraan de client-computers kunnen verbinden en een computer waar we de nodige server-software op installeren.



Afbeelding : netwerk architectuur

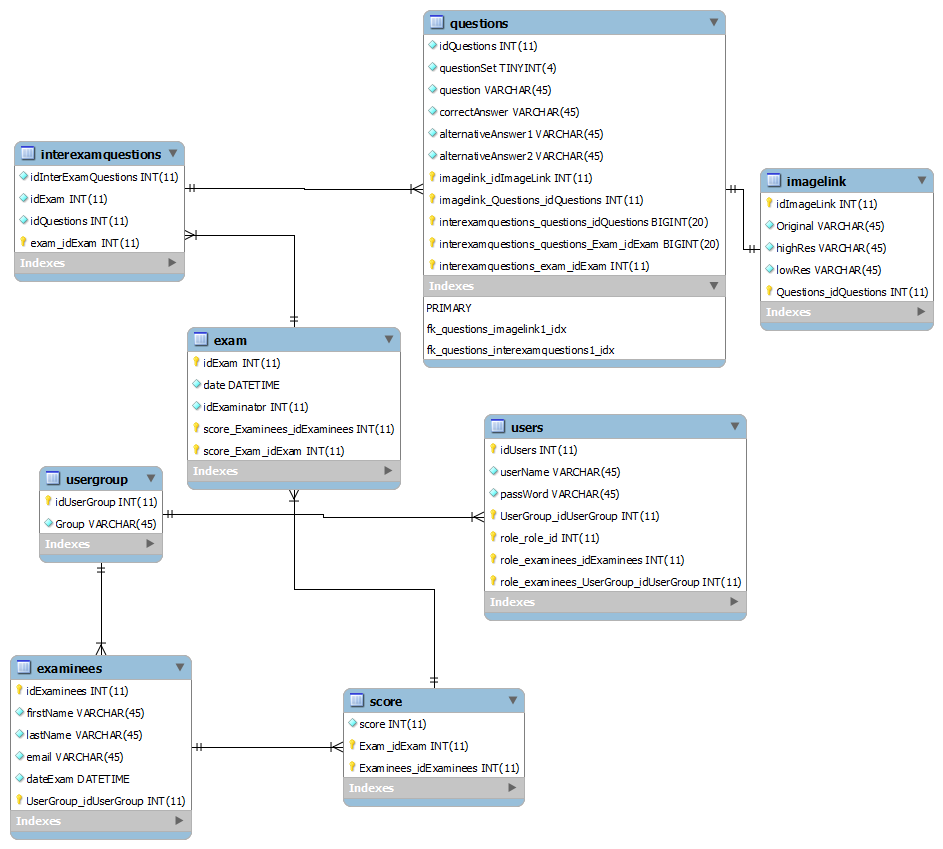


# Software beschrijving

## Database (FH)

Als Database hebben we gekozen om te werken met MySql als databasesoftware. Deze relationele database wordt heel vaak gebruikt, is stabiel en ik makkelijk uitbreidbaar voor alle mid-size toepassingen.

Als bewerkings en editing-tool gebruiken we MySQL Workbench 6.0. Op een eenvoudige grafische manier is het mogelijk om, gebruik makend van de Crow’s foot notering, de relaties tussen de verschillende tabellen vast te leggen. Ook de PK en FK en andere kolomkoppen met hun types worden duidelijk weergegeven.



Afbeelding : EER diagram

Vetrekkende van de tabel ‘usergroup’ komen we bij 2 type gebruikers. ‘Users’ zijn beschemd met een hash-paswoord en vormen de examinators en admins. ‘Examinees’ leggen het examen of de quiz af. Deze tabel heeft een duidelijk andere functie.

Deze laaste is gelinkt aan ‘score’. Elke examinee kan verschillende keren het ‘exam’ afleggen en deze heeft telkens een ‘score’. Elk examen is samengesteld uit een poule van examen vragen (‘questions’) die door de examinator worden gebundeld in een tussentabel ‘interexamquestions’.

Elke vraag heeft een verwijzing naar 1 afbeelding ‘imagelink’. Deze afbeeldingen worden niet in de database opgeslagen, slechts enkel een verwijzing ernaar. Er bestaan verschillende versies van hetzelfde bestand, nl een Origineel, een hoge resolutie en een lage resolutie om naar het LED paneel te sturen.

## Front-end

### Zend Framework (JVW)

Voor de admin en examinator pagina's, hebben we gekozen om deze te verwerken in het Zend Framework. Het gebruik van een framework geeft ons bepaalde voordelen inzake de development en debugging. Op de website van het Zend Framework, staan volgende voordelen vermeld, die we zeer zeker kunnen beamen:

* Modulair : Het zend framework bestaat eigenlijk uit allerlei modules, die elk geconfigureerd en aangepast kunnen worden naar de specifieke eisen van dat bepaalde deel. Anderszijds is het ook een groot pluspunt dat de modules onafhankelijk werken: met andere woorden, een team kan elk aan een stuk werken, elk zijn eigen pagina's maken, elk zijn eigen stuk debuggen en dan alles samenvoegen als één geheel. Onafhankelijke stukken blijven ook onafhankelijk. Als er een deel ontwikkeld en getest is, kunnen we zonder problemen een ander stuk gaan implementeren, zonder dat de werking van het eerder gemaakte stuk in het gedrang komt (of wordt tenietgedaan).

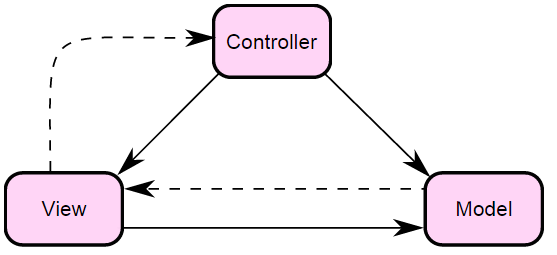
* Uitbreidbaar: Het framework zelf voorziet al in heel veel vooraf gemaakte stukken. Een paar voorbeelden zijn databaseverbindingen, sql-objecten, sessions, serializers, etc. Elk van deze zit gecentraliseerd in de map van de vendor, maar toch kan alles worden aangepast. Als bijvoorbeeld de session niet werkt zoals gewensd, is het “eenvoudig” om deze klasse aan te passen tot ze specifiek bij de eigen eisen past. Hoewel, het aanpassen van zo'n klasse, kan wel wat werk in beslag nemen, zeker ook omdat vele klassen met elkaar gelinkt zijn. Het kan dus nodig zijn om meerder objecten aan te passen.
* High Performing: Elk stuk van het ZF is geoptimaliseerd om zo snel mogelijk te werken, maar toch een zekere security (zie volgende punt) te bewaren. Op alle delen van het ZF is er een fine tuning uitgevoerd om timing te halen en latency te verkleinen.
* Secure: Security is de dag van vandaag een hele boterham om te behouden. Zend steekt heel veel energie in het zorgen voor een vrij grote security. Hashing met een salt is mogelijk, sql-injecties worden (zo veel mogelijk) tegengegaan en er zijn constant updates van het framework, zodat de veiligheid van de applicatie ten alle tijde zo groot mogelijk is.
* Community: Er is een vrij grote community van mensen die met Zend werken. Zodoende is er ook een voortdurende interactie tussen de noden van de gebruikers en de implementatie van de developers. Ook de onderlinge user community is een groot pluspunt, zeker voor iemand die net begint met programmeren in het ZF.
* Enterprise Ready: Zend Framework wordt ook gebruikt door verschillende bedrijven zoals Cisco Webex, PNB Paribas en CNN. Ook zij kunnen naar Zend rapporteren over bijvoorbeeld bugs of eventuele verbeteringen, alsook voor nieuwe features.

Echter, een zeer groot nadeel van Zend, is dat het als beginner zeker niet gemakkelijk is om het “zomaar” onder de knie te krijgen. Een kennis van web-development en het MVC-model is zeker nodig. Om dit beter op te vangen, heeft Zend ook een tutorial gemaakt, maar deze is zeker niet toereikend om te kunnen zeggen dat iemand het volledige framework al onder de knie heeft. Alle documentatie is wel te vinden op hun eigen website, dus vanuit dat oogpunt is Zend zeker toegankelijk om te leren, hoewel er veel tijd inkruipt.

Een van de grootste voordelen voor developers in het Zend Framework is dat voor bijna alle uitvoeringsfouten een foutmelding is voorzien. Dit maakt het zeer gemakkelijk om te debuggen. Ook kunnen de fouten kunnen worden opgevangen, waardoor de gebruiker de specifieke foutmeldingen niet te zien krijgen, maar een zelfgemaakte error pagina.

### MVC-model (JVW)

Het Zend Framework is geïmplementeerd met het MVC-model. MVC staat voor Model, View en Controller.



Afbeelding : MVC model

De Model bevat de klassen en de layout van de objecten die worden gebruikt. In dit project bijvoorbeeld, is er een model voor de admins en examinators, die worden gemodelleerd als gebruikers.

De controller bevat (vrijwel) alle logica die de applicatie gebruikt en reageert op events. Voor dit project zijn de events meestal submit- of andere buttons.

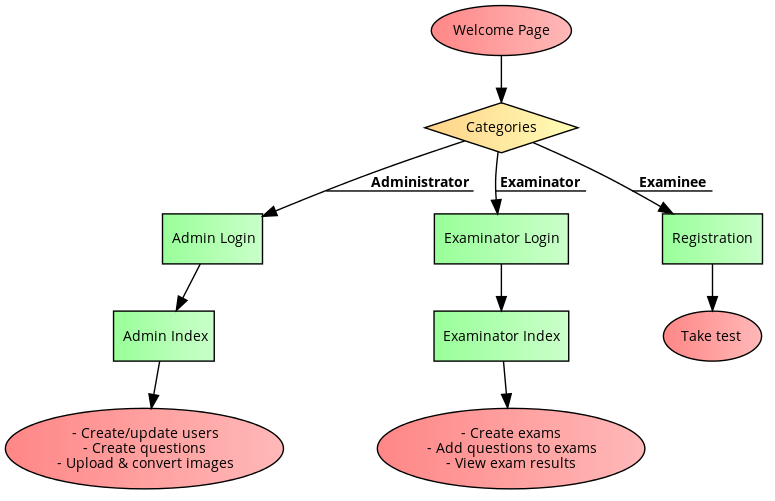
De view beheert alle pagina's die worden getoond naar de gebruikers. Deze kan ook eventueel wat logica bevatten, maar logica die specifiek van toepassing is op de view.

Dit project maakt gebruik van zowel de model, view als controller. De volledige implementatie van het in- en uitloggen van gebruikers is gebaseerd op dit model. Zoals het volgende stuk zal aangeven, zitten al de acties verwerkt in de twee hoofdcontrollers, die van de admin en de examinator. Ook de naamgeving in Zend is zeer transparant. Wordt er bijvoorbeeld geklikt op de logout-button, dan zal de controller de logoutAction uitvoeren, met de logout.phtml view. Hierdoor is het heel gemakkelijk om bij te houden (ook in de debugging) waar het eventueel fout loopt en waar de applicatie zich bevind in de uitvoering.

### Applicatie layout (JVW)

Zoals eerder vermeldt, zit bijna alle logica in de controllers. De onderstaande figuur toont de layout voor de applicatie. Daarin is te zien hoe alles mooi via de controllers kan verlopen, zowel het laden van de index pagina, als het doorgaan naar andere acties. De controller zorgt er voor dat elke view aan de juiste actie wordt gekoppeld. Wil een admin bijvoorbeeld een nieuwe gebruiker toevoegen of een bestaande aanpassen, dan gaat dit allemaal via de admincontroller passeren.

Voor twee bepaalde acties springt de applicatie buiten het ZF. Dit is wanneer we afbeeldingen toevoegen aan de vragen (Admin->uploadImages) en wanneer de examinee de test begint. Het registreren van de examinees gebeurt nog binnenin het ZF (omwille van de beveiliging die ZF biedt).



Afbeelding : framework workflow

Zoals bovenstaande afbeelding laat zien, is er gezorgd voor een duidelijke structuur in de applicatie. Zowel de admin als de examinator moet inloggen op de applicatie. Bij de login wordt gecontroleerd welke rol de gebruiker heeft (admin of examinator). Indien een gebruiker met een bepaalde rol op een verkeerde inlogpagina inlogt, zal hij met een succesvolle login toch verwezen worden naar zijn eigen index pagina. Met andere woorden: als een admin probeert in te loggen op de examinator login, zal hij met succes kunnen inloggen, maar wordt doorgestuurd naar de admin index pagina. Het omgekeerde geldt natuurlijk voor de examinator.

De functies van de admin zijn nog vrij beperkt, dit is nog een punt voor verbetering indien er wordt verdergewerkt aan het project. Voorlopig kan de admin gebruikers toevoegen, zowel admins als examinators. Hun rol kan worden gespecifieerd bij het aanmaken.

Het wachtwoord is voorlopig nog een simpele hash. Dit kan worden uitgebreid met een salt, alsook met een sterkere hashing function. Voor het concept is er geopteerd voor een md5, maar er zijn natuurlijk sterkere functies beschikbaar in PHP (zoals de SHA-256 of SHA-512). De md5 wordt ook in de database opgeslagen, zodat er geen “clear text” wachtwoorden zichtbaar zijn in de databank. Een admin kan de rol van een gebruiker wijzigen, het wachtwoord echter niet. Omwille van tijdsgebrek de functie voor wachtwoorden te resetten of opnieuw in te stellen niet toegevoegd.

Een andere verantwoordelijkheid van de admin is het toevoegen van vragen. Deze vragen zijn kunnen algemeen zijn, en kunnen later door de examinator toegevoegd worden aan het examen. Ook voor het toevoegen van een bepaalde afbeelding is de admin gemachtigd. De applicatie zelf neemt de conversie voor z'n rekening, alsook het linken van de vraag met de juiste afbeelding in de database.

De examinator heeft een heel andere set van mogelijkheden. Op de indexpagina worden alle examens weergegeven die een examinator ooit heeft gegeven. Hij ziet echter alleen maar z'n eigen examens, niet die van andere examinatoren. Bij elk examen kan hij de resultaten opvragen. Bij de resultaten worden alle examinees opgelijst die aan dit specifiek examen hebben deelgenomen. Er wordt weergegeven wat hun naam en hun behaalde score op het examen was.

De examinator kan een nieuw examen aanmaken, en daarbij klikken op de toe te voegen vragen. Eens een examen is gemaakt, kan dit niet meer worden aangepast (functie wegens tijdsgebrek niet geïmplementeerd). Het nieuwe examen is enkel voor hem toegankelijk. Elke examinator zal dus een eigen lijst van examens hebben. Onderling uitwisselen van examens wordt hiermee geblokkeerd. Dit werd zo bepaald omdat specifieke examinatoren specifieke vragen kunnen stellen. De ene examinator is bijvoorbeeld iemand die examens geeft over verkeer, terwijl een andere heel algemene vragen kan stellen op z'n “examen”, denk bijvoorbeeld aan een quiz zoals blokken. De termen veranderen, maar uiteindelijk is het een multi-functionele quiz-computer, niet enkel bedoeld om examens af te leggen over verkeersborden. Het is dus puur een kwestie van design decisions.

Voordat een examinee kan beginnen aan een examen, moet er een examen op actief worden gezet. Dit is de taak van de examinator. Als er geen actief examen is gezet, wordt er in plaats van een foutmelding, een gewone error pagina weergegeven. Daarop staat te lezen dat er een fout is opgetreden en dat de examinee de examinator moet verwittigen. In de handleiding van de applicatie zal er natuurlijk staan wat het probleem is van deze melding.

### Specifieke pagina's (JVW)

Alle bovenstaande functies voor admin en examinator beginnen vanaf de index pagina. De keuze hiervoor was om alles centraal te houden en een overzichtspagina te hebben.

Op de indexpagina van de admin, staat er een overzicht van de gebruikers in een tabel. Daarin kan de functie die ze bekleden worden gezien, alsook staat de link om een gebruiker aan te passen of te verwijderen in dezelfde tabel (als laatste twee kolommen). Onder de gebruikstabel is de vragentabel te zien. Deze tabel bevat de vragen en antwoorden, met net zoals in de gebruikstabel een link om de vraag te wijzigen. Het id van de vraag kan niet worden gewijzigd, de vraag zelf wel. Wegens tijdsgebrek is er geen functie om een vraag te verwijderen en wordt de afbeelding bij het updaten van een vraag niet weergegeven. Er is dus nog redelijk wat ruimte voor verbetering.

Als een examinator inlogt, ziet hij de examinator indexpagina. Op deze pagina krijgt hij een overzicht van alle examens die hij ooit heeft aangemaakt en waar eventueel examinees hebben aan deelgenomen. Onderscheid tussen examens met of zonder examinees wordt niet gemaakt. Voor elk examen kan daarvan een overzicht worden aangevraagd. Op deze pagina staat ook een directe link om een examen op actief te zetten. De gedachte hierachter is dat, wanneer de error pagina wordt weergegeven, er niet teveel tijd verloren gaat om een examen actief te zetten. De examinator heeft dus maar in te loggen en kan meteen het examen activeren. Wanneer de resultaten worden weergeven, ziet de examinator de gegevens van de examinee, zoals al eerder vermeldt.

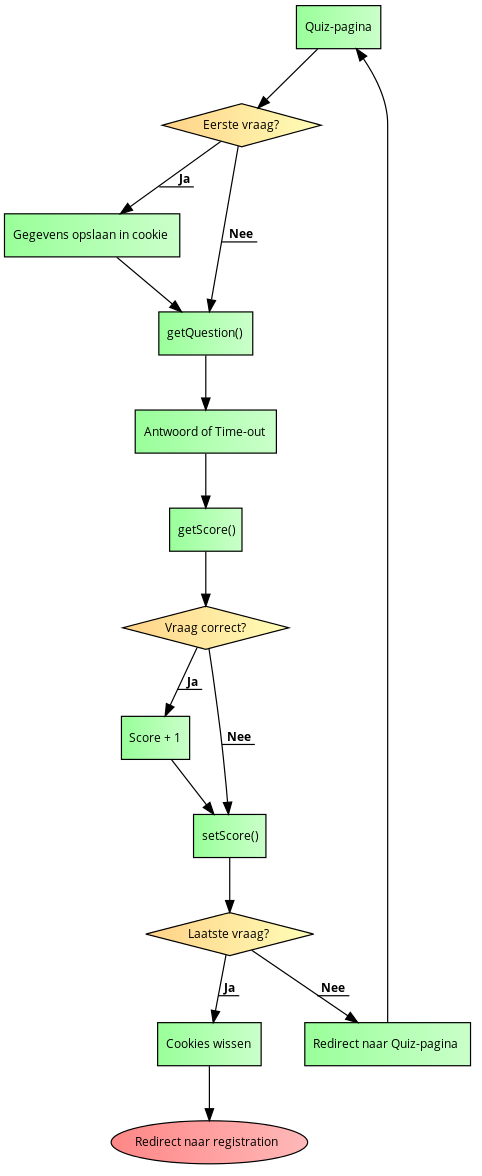
Het aanmaken van een nieuw examen gebeurt met de daarvoor bestemde link. Bij een succesvolle creatie van een examen, wordt de examinator meteen doorverwezen naar de pagina waar alle vragen worden opgelijst. Daar heeft de examinator maar te klikken op de “voeg vraag toe”-link in de tabel. Er werd vooral geopteerd om een intiutieve applicatie te schrijven. De functies worden aangeroepen door middel van een muisklik.

### De Quiz-pagina (GVB)

De quiz-pagina is één van de belangrijkste pagina's van de web-applicatie. Het doel van de quiz-pagina is overduidelijk: vragen stellen aan de gebruiker en hun antwoord registreren om een score te bepalen. Op de volgende pagina staat een kort overzicht van de werking van deze pagina. Nadat de gebruikers zich geregistreerd hebben, onthouden we enkele variabelen voor gebruik in de quiz-pagina. In een PHP-sessie onthouden we het ID van het actieve examen en de reeks van vragen. Bij het doorsturen naar de quiz-pagina geven we met GET-variabelen het e-mailadres van de gebruiker mee en een variabele die beslist of er afbeeldingen naar de mbed mogen gestuurd worden. Tijdens het uittesten van de applicatie was er immers een probleem waarbij elke gebruiker die een quiz aan het oplossen was, zijn afbeelding naar de mbed stuurde. Dit had als gevolg dat er om de paar seconden een nieuwe afbeelding naar de mbed gestuurd werd. Daarom zal deze nieuwe variabele de waarde “yes” of “no” hebben om duidelijk te maken of deze gebruiker zijn afbeelding mag sturen of niet.

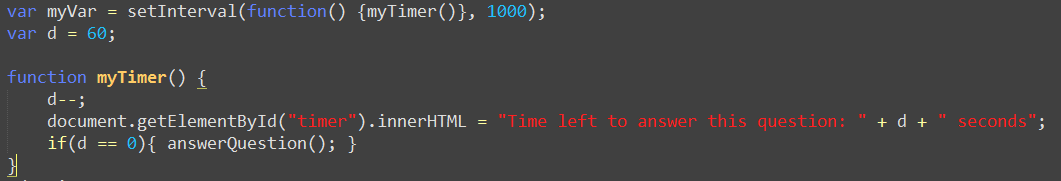
Wanneer de quiz-pagina laadt, zal er gecontroleerd worden of dit de eerste keer is. Dit gebeurt door een controle op de aanwezigheid van een cookie. Als er geen cookie aanwezig is, dan is het de eerste keer, en zullen we de cookie aanmaken. In de cookie zullen we de verschillende waarden bijhouden uit de SESSION – en GET – variabelen. Daarnaast voegen we ook de index van de vragenlijst toe, dat nul zal zijn wanneer de pagina voor het eerst laadt. Als er wel een cookie aanwezig was, verhogen we de waarde van de vragenindex met een. Nadat we deze waarden heb opgeslagen voor verder gebruik, halen we de vraag uit de database die bepaalt wordt door de vragenlijst en de index. Op de server zoeken we in de database naar de vraag, antwoorden en afbeeldingen die bij het ID van de vraag passen. Als we geen resultaat terugkrijgen, wil dit zeggen dat er bij deze vraag geen afbeelding hoort. Dan voeren we een tweede query uit waarbij we alleen de vraag en antwoorden ophalen en terugsturen.

Als er wel afbeeldingen zijn, voeren we op de server nog enkele berekeningen uit voordat we de data terugsturen. Indien de gebruiker naar de mbed mag sturen, zullen we de 32x16 versie van afbeelding in geheugen maken en pixel per pixel de afbeelding overlopen. We bepalen de kleur van elke pixel en afhankelijk van de rgb-waarde maken een byte die de kleur voorstelt. Daarna sturen we een byte-array met de kleuren van alle pixels naar de mbed. De webpagina zelf heeft de vraag en antwoorden ontvangen en eventueel het pad naar de afbeelding die bij de vraag hoort.



Afbeelding : Question Workflow

De antwoorden worden in willekeurige volgorde weergegeven in een tabel. Dit doen we aan de hand van een array die de ID's van de elementen bevat waar we de antwoorden in weergeven. Met een shuffle-functie komen de elementen in een willekeurige volgorde. Daarna kennen we het juiste antwoord toe aan het eerste element in de array, de foute antwoorden aan de andere elementen in de array. In een verborgen veld naast de antwoorden houden we bij welke van de antwoorden de correcte was. Dit is misschien de simpelste en onveiligste manier, maar we gaan ervan uit dat de deelnemers niet naar de broncode van de pagina kunnen kijken om na te gaan wat het correcte antwoord is. Naast de vraag, antwoorden en eventueel een afbeelding bevat de pagina ook een timer om de gebruikers te beperken in tijd die ze hebben om een vraag te beantwoorden.

Dit doen we aan de hand van het volgende stuk code:

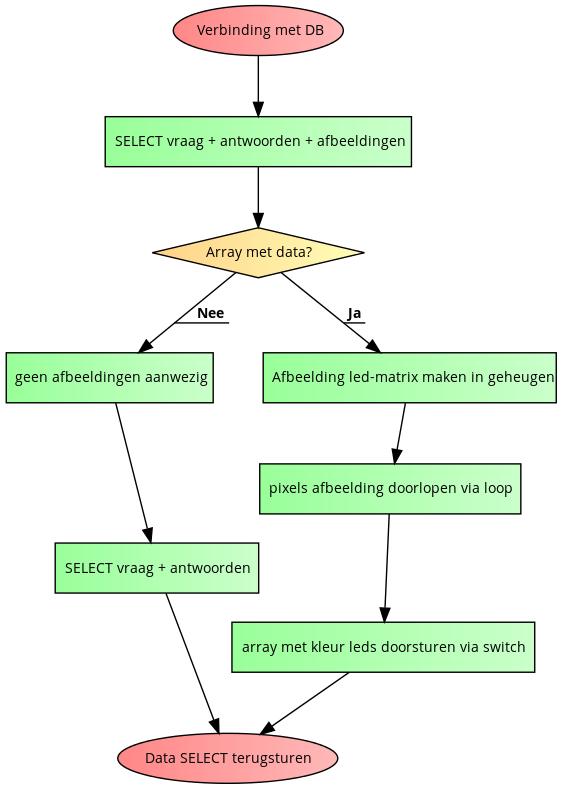
Afbeelding : codefragment Timer

Aan de hand van 'setInterval' zullen we iedere seconde de functie 'myTimer' uitvoeren. We beginnen met de timer op 60 seconden. Wanneer we onze functie aanroepen, zullen we dit getal verminderen en het nieuwe resultaat weergeven op het scherm. Als ons getal nul bereikt, zullen we automatisch de functie uitvoeren om de vraag te beantwoorden. Omdat onze timer in javascript is en dus client-side kan de gebruiker in principe de timer manipuleren maar dit lijkt weinig waarschijnlijk.

De gebruiker kan natuurlijk zelf naar de volgende vraag gaan door op de knop te drukken om te antwoorden. Wanneer de functie uitgevoerd wordt om te antwoorden, controleren we eerst of er een antwoord aangevinkt is en of dit het juiste antwoord is. Daarna halen we de score uit de database van de huidige gebruiker. Als het antwoord op de vraag juist was, verhogen we de score, anders blijft de score hetzelfde. Daarna sturen we de score terug naar de database. Het leek een beter idee om per vraag de score aan te passen in tegenstelling tot de score bij te houden en pas na de laatste vraag door te sturen naar de database. Indien een gebruiker de verbinding verliest, zal er toch een score zijn voor de vragen die wel zijn opgelost.

Na het doorgeven van de score controleren we of dit de laatste vraag was of niet. We nemen de index van de vragen en de array met vragen. Als de index plus één gelijk is aan de lengte van de vragenlijst, dan was dit de laatste vraag. In dat geval wissen we alle cookies met informatie en sturen we de gebruiker terug naar de registratie-pagina. Als het niet de laatste vraag was, laden we quiz-pagina gewoon opnieuw. Daardoor hebben we opnieuw de controle of de cookies al bestaan, …

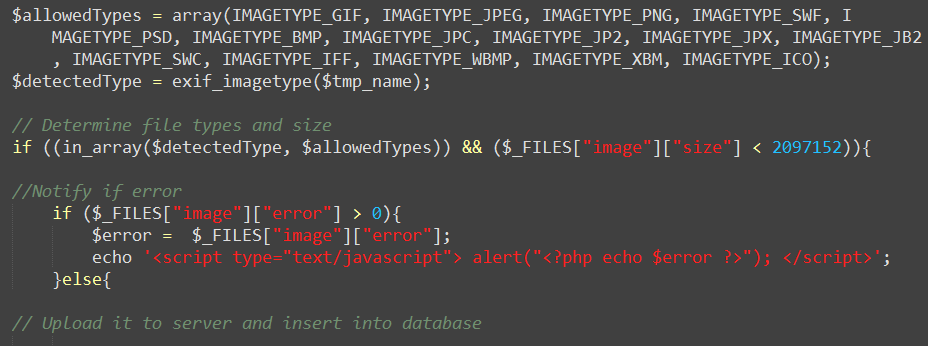
Overzicht afhalen van de vraag:



Afbeelding : communicatie met Dbase

### Uploaden van afbeeldingen

Op de index-pagina van de administrators kan men o.a. de lijst met alle vragen zien. Naast elke vraag is er een knop om een afbeelding toe te voegen aan deze vraag. Door op de knop te drukken komen we op de upload-pagina. Om duidelijk te maken voor welke vraag we een afbeelding gaan toevoegen, geven we het ID van de vraag mee in een GET-variabele. In de tabel in de database voor het opslaan van de afbeeldingen gaan we enkel het pad naar de afbeeldingen opslaan in tegenstelling tot de afbeelding zelf. Databases zijn dan wel in staat om afbeeldingen op te slaan, maar het is veel efficiënter als de afbeeldingen op de server staan. In onze applicatie zou er waarschijnlijk niet zoveel verschil zijn omdat we maar een beperkt aantal afbeeldingen hebben, maar we hebben toch besloten om het op deze manier te implementeren.

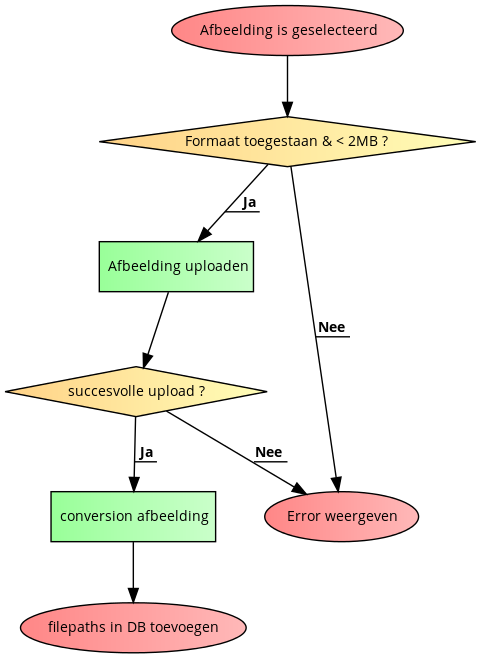
Op de upload-pagina kunnen de administrators het bestand selecteren dat ze willen toevoegen. Bij het uploaden controleren we eerst of het geselecteerde bestand een afbeelding is. Dit doen we aan de hand van de volgende code:

Afbeelding : image upload

We hebben een array met types die overeenkomen met afbeeldingen. Daarnaast hebben we het type van het bestand dat geselecteerd was. Als het type van het bestand voorkomt in de lijst van toegestane types, dan zijn we zeker dat het een afbeelding is. Zo niet dan geven we een waarschuwing weer.

Er is niet alleen een beperking op type, maar ook op grootte van het bestand dat men wil toevoegen. De afbeelding moet kleiner dan 2 MB zijn om te mogen uploaden. Als de afbeelding groter is, wordt er ook een waarschuwing weergegeven.

Als de afbeelding voldoet aan deze vereisten, dan wordt het op de server opgeslagen. Nadien maken we twee alternatieve versies van de afbeelding. We maken een 300x300 versie de we gaan gebruiken om weer te geven op quiz-pagina. Daarna maken we nog een 32x16 versie die gebruiken om naar de mbed te sturen.

Wanneer de drie versies van de afbeelding opgeslagen zijn op de server, voegen we in de database de paden van deze afbeeldingen toe. Een overzicht van het uploaden kan je hieronder zien:

Afbeelding : workflow Image upload

## Serversoftware

Francis

## Stuurprogramma Microcontroller

Kevin

# Toekomstig Werk

## Beveiliging

Francis

## Compatibiliteit verschillende browsers

Jonas & Gerd

# Conclusie

All