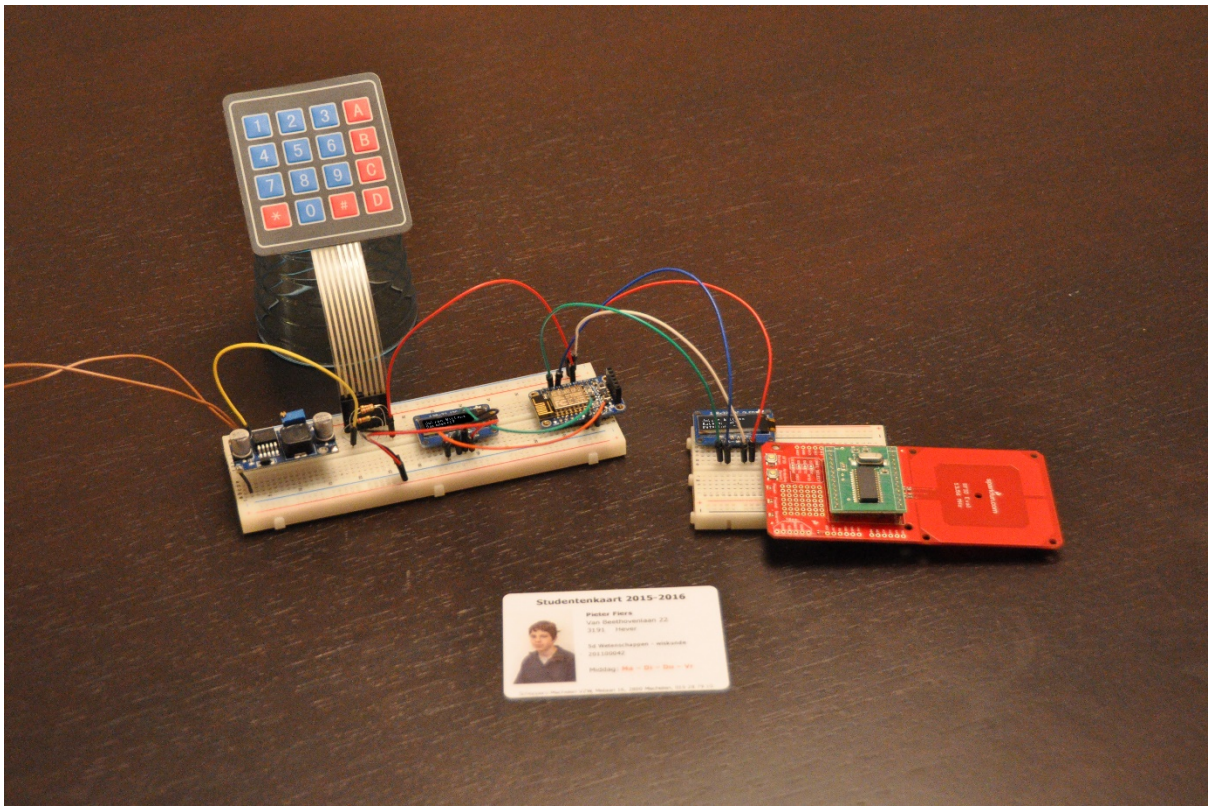
Digitaal betaalsysteem op school



DSpay

Auteur: *Pieter Fiers, 6C WeWi*

School: *ASO Scheppers Mechelen*

Schooljaar: *2016-2017*

Promotor: *Meneer Moens*

Vakgroep: *Wetenschappen*

Onderzoeksvraag: *Hoe implementeer ik een digitaal betaalsysteem op school?*

Inleiding

Schepperianen die een warme maaltijd of het gebruik van de printer moeten betalen doen dat door contante betaling. Nochtans heeft onze school vroeger digitale systemen gebruikt voor het betalen van kleine transacties op school. Er waren inderdaad sinds begin de jaren 2000 specifieke betaalsystemen van Xafax en Actips in gebruik, elk met zijn voor- en nadelen.

Met dit eindwerk heb ik als ambitie om een beter betaalsysteem te ontwerpen dan eender welk vorig systeem dat de school al heeft gebruikt.

Dit eindwerk is in drie delen verdeeld:

In het eerste deel A onderzoek ik de voor- en nadelen van verschillende systemen, al dan niet ooit door Scheppers gebruikt.

In deel B verwerk ik die informatie tot een beeld van hoe het ideale betaalsysteem zou werken. Dit doe ik door eerst vanuit het perspectief van de verschillende gebruikers van het systeem te kijken en daarna door de verschillende functies die het systeem nodig heeft te bepalen en samen te brengen.

In het laatste deel C tenslotte breng ik de theorie in de praktijk door een aantal onderdelen ook echt te implementeren in een proef-concept, van de hardware voor de betaalterminals tot de software die het systeem bestuurt.

Aangezien ik vaak gebruik maak van technische termen heb ik op het einde een verklarende woordenlijst toegevoegd.

Dankwoord

Bij het afleveren van dit eindwerk wil ik graag nog enkele mensen bedanken: meneer Van Langenhove voor het verspreiden van de enquête via de nieuwsbrief van het Scheppersinstituut; de vele ouders die de enquête hebben ingevuld; directeur Marc Vercauteren voor de antwoorden op mijn vragen over vorige betaalsystemen en mijn promotor, meneer Moens voor de vlotte begeleiding en feedback. Tot slot, mijn papa voor het nalezen van mijn eindwerk en het sponsoren van de hardware om het prototype te ontwikkelen.

Inhoudsopgave

Deel C:

Deel A:
Analyse van bestaande systemen

In dit hoofdstuk onderzoek ik de voor- en nadelen van enkele betaalsystemen. Ik bespreek het vorige en ook huidige betaalsysteem dat op school wordt gebruikt. Er komen ook enkele interessante prototypes aan bod. Ik leg ook de technologieën uit die door de verschillende systemen gebruikt worden.

1. Vorige systemen

Ik kijk eerst terug op eerdere systemen die Scheppers heeft gebruikt, de reden dat ze niet meer in gebruik zijn en wat er aan die systemen kan verbeterd worden. Als informatiebronnen voor deze bespreking gebruikte ik mijn interview met de heer Vercauteren aangevuld met enerzijds mijn eigen ervaringen met de betalingssystemen en anderzijds die van mijn zus en broer (oud-Schepperianen).

1.1. Xafax

Over systemen die de school voor 2000 gebruikte heb ik niet veel informatie kunnen verzamelen, behalve het feit dat men waarschijnlijk werkte met contante betaling, zoals nu. In 2000 implementeerde de school het allereerste digitale systeem. Leerlingen konden contact- en contantloos betalen voor de warme maaltijd en het gebruik van de printer. Dit systeem werd onderhouden en geïmplementeerd door Xafax. Xafax, toen een in de Benelux alomtegenwoordig bedrijf, installeerde de betaal- en een herlaadstations om leerlingen via contante betaling krediet op hun betaalsleutel te laten zetten. Xafax onderhield dit systeem en programmeerde de sleutels. Tegenwoordig doen ze dat alleen nog voor scholen en bedrijven in Nederland. Wat Xafax niet zelf ontwierp en produceerde waren de betaalsleuteltjes en hun digitale lezers. Deze zogenoemde *U-keys* en hun lezers waren een ontwerp van het Zwitserse bedrijf Microtronic. Microtronic ontwikkelde toen en nu nog steeds verschillende soorten U-keys van sleuteltjes tot betaalkaarten. Deze kleine sleutels en kaarten zijn zoals de meeste digitale identificatie-systemen gebaseerd op RFID en meer bepaald NFC, een onderverdeling hiervan. Om de U-key te bespreken moeten we dus eerst weten wat RFID en vervolgens ook NFC precies inhouden.

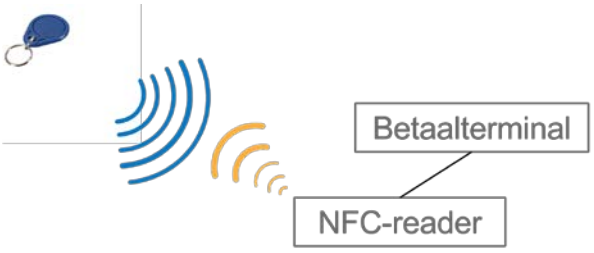


Fig. 1: Illustratie van NFC.

RFID, of radio-frequency-identification, is een door de ISO/IEC JTC 1 [1] gestandaardiseerde technologie[2] voor RFID-tags en -lezers (vanaf nu *readers* genoemd om de vakterminologie te respecteren) en hun communicatie. RFID-tags en hun data kunnen door een RFID-reader met antenne worden uitgelezen. Zowel de tag als de reader bevatten dus een stroomvoorziening en een radio-transceiver. De tags hebben daarnaast ook een klein geheugen voor het opslaan van een id. Ze kunnen passief of actief[3] zijn afhankelijk van hun stroomvoorziening. Actieve tags hebben een eigen stroomvoorziening in de vorm van bijvoorbeeld een kleine batterij terwijl passieve tags hun stroomvoorziening krijgen in de vorm van de radiosignalen van de RFID-reader, volgens het principe van elektrische inductie. Passieve RFID is een voorbeeld van een master-slave communicatiemodel aangezien de master (RFID-reader) de voeding voorziet voor het apparaat waarmee die communiceert; de slave (RFID-tag). Daarnaast is het de reader die data opvraagt van de tag, dus geen wederzijdse communicatie. De communicatie

[1] (Rajchel, n.d.)
[2] (ISO/IEC, 2013)
[3] (Thrasher, 2013)

- en dus bij passieve tags ook stroomvoorziening - tussen tag en reader verloopt over radiogolven van Low Frequency (LF) 125 - 134 kHz, High Frequency (HF) 13.56 MHz of Ultra High Frequency (UHF) 856 - 960 MHz.[4] RFID technologie wordt vaak toegepast voor de identificatie van bv. pakketjes in een opslagplaats of dieren in het wild.

NFC, of near-field-communication is een standaard voor dataoverdracht gebaseerd op HF-RFID, dus 13.54 tot 13.56 MHz. NFC verschilt van RFID in de zin dat RFID slechts voor identificatie ontworpen is, terwijl men met NFC relatief grotere hoeveelheden data kan overdragen. NFC heeft net als RFID een actieve en een passieve variant. Actieve NFC vindt men vooral in smartphones waarbij beide transceivers gevoed worden door de smartphones. Met zo'n actieve communicatie kan men via standaarden zoals Android Beam of Apple's Airdrop bestanden of contacten overzetten. Hoe dit precies werkt is nog iets geavanceerder dan gewoon NFC-communicatie, daar zal ik dan ook niet verder op ingaan. De meest gebruikte vorm van NFC, en degene waar ik het in dit eindwerk vooral over ga hebben, is passieve NFC. Hierbij krijgt de NFC-tag dus zijn voeding van de reader en vraagt de reader over HF-radiogolven de benodigde data op. Dat de NFC-tag hierbij geen eigen voeding nodig heeft is vooral handig voor bijvoorbeeld kleine betaalsleutels of ID-kaarten.

Een van de belangrijkste producenten van NFC-tags voor betaal- en toegangssystemen is het Nederlandse NXP Semiconductors, vroeger een divisie van Philips. NXP's belangrijkste lijn NFC producten zijn de zogenoemde MIFARE-chips. Het zijn ook deze soort chips die Microtronic gebruikt voor hun U-keys en tags. Microtronic's implementatie van de MIFARE-chips heeft een specifiek manier voor het opslaan van data met informatie over bijvoorbeeld de gebruiker van de sleutel en de geldigheidsdatum. De specifieke manier van opslag is een industrieel geheim en is uiteraard beveiligd tegen wijziging.

Xafax[5] (eerste cashloos betaalsysteem op Scheppers) gebruikte Microtronic's implementatie van NXP's passieve NFC technologie. Hoewel alle benodigde data voor een transactie technisch gezien al op de U-key zou kunnen staan moeten alle betaalstations in een netwerk van Xafax ter beveiliging toch met elkaar verbonden zijn. Dit gebeurt via een specifiek bedraad netwerk dat de mogelijkheid geeft om bv. bij diefstal sleutels te blokkeren of transacties na te kijken. Het grote nadeel van dit systeem is uiteraard dat het niet altijd even makkelijk is om een draad te leggen langs alle betaal- en herlaadstations. En hoewel alle stations (vanaf nu terminals, om de vakterminologie te respecteren) met elkaar konden communiceren had Xafax' systeem geen optie om bv. geld over te schrijven naar de rekening van een leerling. De enige manier om geld op een sleutel te krijgen was - net zoals bij het volgende systeem dat ik zal bespreken - via een oplaadautomaat.

1.2. Actips

Scheppers werkte lange tijd met Xafax' systeem. Maar wegens het duurder wordende onderhoudscontract en het verouderen van het in 2000 geplaatste Xafax systeem werden andere opties aantrekkelijk. Daarom schakelde de school over naar het contactloos betaalsysteem van Actips. Actips, gevestigd in Sint-Katelijne-Waver gebruikte een andere implementatie van zowel de betaalterminals als van de betaalsleutels. De betaal- en herlaadterminals in Actips' systeem zijn niet onderling verbonden. Aangezien functies zoals live-monitoring geen prioriteit waren in de ogen van de school was dit een groot voordeel ten opzichte van het bedrade Xafax systeem. Het was zelfs nog steeds mogelijk om bv. sleutels te blokkeren of het saldo van een bepaalde gebruiker na te kijken. Dit gebeurde dus niet via een bedraad netwerk maar door fysiek verbinding te maken met een van de terminals zelf. De terminals hielden informatie bij over alle vorige transacties en konden geprogrammeerd worden om bepaalde sleutels te weigeren.

Die sleutels waren hoogstwaarschijnlijk ook een variatie op de MIFARE passieve NFC-tags, maar dit is moeilijk te bepalen aangezien Actips zelf niet meer actief is. Ze werden door het Belgische bedrijf ESE[6] overgenomen. De

[4] (Impinj, s.d.)
[5] (Xafax, s.d.)
[6] (ESE, s.d.)

meeste van ESE's betaalsleutels gebruiken inderdaad MIFARE NFC-chips, meer bepaald de MIFARE Ultralight chips, die specifiek bedoeld zijn voor toegangs- en betaalsystemen. Wat wel vaststaat is dat de Actips betaalsleutels met een universelere datastructuur werkten dan de U-key, daarom konden die sleutels ook bij andere producenten aangekocht worden. Dit wijst er nog eens op dat Actips waarschijnlijk MIFARE-chips gebruikte.

Maar zoals eerder gezegd was Actips' systeem ook geen lang leven beschoren. Bij de overname door ESE verloor Actips de werknemer die onder andere instond voor het onderhoud van het betaalsysteem op Scheppers. Omdat ook het onderhoudscontract wijzigde, werd er door Scheppers beslist om het contract met Actips twee jaar geleden, in 2015, te verbreken. Er werd geen nieuw systeem, zoals bv. dat van Antenor (besproken in 'Alternatieve systemen') geïnstalleerd.

1.3. Besluit

Het belangrijkste doel van deze bespreking is om te leren uit de positieve en negatieve kanten van vorige systemen. Daarom even een korte samenvatting van de voor- en nadelen van Xafax' en Actips' systeem.

Het betaalsysteem van Xafax had als voordeel dat transacties bekijken en sleutels blokkeren makkelijk kon zonder met elke betaalterminal te moeten connecteren. De belangrijkste nadelen waren dat er een draad gelegd moest worden tussen alle terminals en dat er geen mogelijkheid was om de sleutel via bijvoorbeeld overschrijving op te laden.

Actips' systeem loste het bedradingsprobleem op met terminals die afgelezen konden worden door er fysiek mee te connecteren. Het had als nadeel dat sleutels blokkeren wat moeilijker was, en opladen via een overschrijving ronduit onmogelijk (aangezien de terminals geen internetverbinding hadden).

2. Huidige systeem

Zoals in het vorige hoofdstuk is besproken gebruikt Scheppers momenteel geen vorm van digitaal betaalsysteem meer. Leerlingen en leerkrachten betalen nu dus contant (*cash*) voor printergebruik of een warme maaltijd. Maar ook dit soort betaalsysteem wil ik bespreken aangezien het ook zijn voor- en nadelen heeft en dus zeker als inspiratie kan dienen voor een digitaal alternatief. Om het verloop van een betaling met dit systeem te onderzoeken en om te begrijpen wat de ouders van dit systeem vinden heb ik een enquête ontworpen (zie bijlage 2). Deze werd, met dank aan meneer Van Langenhove, via de Scheppers nieuwsbrief en op de schoolwebsite verspreid en, met dank aan mijn moeder, naar de leden van de ouderraad doorgemaild. Daarnaast kan ik ook bevindingen van mijzelf en mijn medeleerlingen over het systeem gebruiken om de voor- en nadelen ervan te onderzoeken.

Naar mijn mening is contant betalen geen ideaal systeem. Ook bij ouders leeft dit idee, aangezien slechts minder dan 8% van de ouders die de enquête invulden vinden dat het systeem echt volledig vlot verloopt. 38,5% van die ouders vinden zelfs dat betalingen niet of helemaal niet vlot verlopen. Hoewel de meerderheid van de ouders toch vinden dat het betalen relatief vlot verloopt, is er dus wel ruimte voor verbetering. Maar waar precies is verbetering het meest nodig? Ik bekijk alle aspecten rond een betaling en haal er de verbeterpunten uit.

Het verloop van een betaling met cash begint eigenlijk al met de leerling thuis geld mee te geven. Als men dit vergeet kan de leerling niet betalen, terwijl met een digitaal systeem bij 'noodgevallen' negatief kan worden gegaan. Voor het betalen van warme maaltijden is hier dan ook een systeem voor bedacht waarbij de naam van de leerling op een briefje wordt geschreven voor toekomstige betaling. Dit is uiteraard ook geen ideale oplossing aangezien een terugbetaling niet gegarandeerd is en er bijvoorbeeld kan vergeten worden om het briefje na te kijken bij de volgende betaling. Zo'n systeem laat ook ruimte voor verschillende interpretaties. Kan een leerling nog een betaling doen als hij/zij al op de lijst staat? Of alleen voor een dessert? Van de ouders die de enquête invulden en vinden dat het huidige betaalsysteem niet vlot verloopt zijn er trouwens meer dan drie kwart die vinden dat dit specifieke probleem de vlotheid ervan ondermijnt. Een belangrijk punt voor verbetering dus!

Maar dit is slechts de voorbereiding op een betaling. De leerling moet dan nog de eigenlijke transactie maken bij de blauwe refter voor een warme maaltijd of dessert, of bij het secretariaat voor het gebruik van de printer. Dat laatste is erg moeilijk aangezien de leerling maar voor kleine sommen geld moet betalen: bijvoorbeeld een taak of een verloren blad uit zijn of haar cursus kopiëren. En met de drukte aan het secretariaat is het niet ideaal om met kleingeld te staan knoeien. Bij de warme maaltijd is dit minder een probleem aangezien het altijd om €4,50 voor een warme maaltijd en 50 cent voor een dessert gaat.

Ten derde is veiligheid een probleem. 45% van de ouders vindt namelijk dat met cash betalen veiliger kan. Geld of een portefeuille 'blokkeren' als die gestolen wordt is niet echt een optie. Een van de ouders merkt hier ook op dat de mogelijkheid om een betaalsleutel te blokkeren een must is voor een digitaal betaalsysteem. Naast dat soort administratie door de school zelf zou een digitaal systeem ook de mogelijkheid geven om bijvoorbeeld transacties na te kijken. Volgens de heer Vercauteren is deze vraag in het verleden door een ouder nog niet gesteld. Maar met een simpele online console zouden ouders het misschien wel op prijs stellen.

Aangezien de meerderheid van de ouders (61,5%) toch vindt dat transacties met het huidige betaalsysteem vlot verlopen zal ik tot slot ook kijken naar de voordelen. Het belangrijkste pluspunt wordt door een van de ouders in mijn enquête opgemerkt: "Ik vind de cash methode goed omdat onze zoon op deze manier een beter zicht heeft op wat hij allemaal uitgeeft." Een fysiek biljet overdragen bij een betaling is krachtiger dan even je studentenkaart ergens tegenhouden. Daarom is het belangrijk om ook bij een digitaal betaalsysteem duidelijke feedback te geven over het succesvolle verloop van een betaling en uiteraard van het betaalde bedrag. In deel B van mijn eindwerk ga ik daar verder op in. Verder moet het nieuwe systeem dus ook een goede beheerdersconsole hebben voor zowel de ouder als de school. Een tweede voordeel van cash betalen is dat 'geld op je rekening zetten' geen moeite is. Makkelijk opladen van de rekening van een leerling is dus ook een must voor een potentieel digitaal systeem.

3. Alternatieve systemen

In dit hoofdstuk bespreek ik kort enkele systemen en oplossingen voor bepaalde aspecten van een zelfontworpen digitaal betaalsysteem. Hiervoor gebruikte ik mijn eigen kennis van microcontrollers en andere technologieën om in een literatuurstudie systemen en prototypen en hun voor- en nadelen te onderzoeken.

Vooraleer ik over onvolledige demo's zal uitweiden bekijk ik eerst de totaaloplossingen van andere bedrijven zoals in het eerste hoofdstuk van dit deel (2.1: Vorige systemen). Dhr. Vercauteren gaf in mijn interview met hem het voorbeeld van het bedrijf Antenor.

Antenor biedt oplossingen aan van geldwisselaars tot volledige digitale betaalsystemen voor scholen, bibliotheken, of kopie-centra. Antenor's hoofdactiviteit is het samenbrengen van technologieën en oplossingen van andere fabrikanten en daarvoor dienstverlening bieden. Een digitaal betaalsysteem voor een school of bedrijf kan bij Antenor met magneetkaarten of een NFC kaart of sleutel werken. Men heeft een breed aanbod aan betaal- en herlaadterminals om hiervan gebruik te maken. Maar ik zal hier niet verder op ingaan aangezien de technologieën die men hiervoor gebruikt (bv. systemen met bankkaart) buiten mijn bereik voor een zelfontworpen systeem vallen. Wat ik wel kan afleiden uit deze systemen is dat voor mijn ontwerp een lage kost mijn sterkste punt is.

Voor mijn toepassing zijn daarom kleine prototypes, eerder dan producten van grote bedrijven, een betere start om ideeën op te doen. Daarom ging ik op zoek naar prototypes met microcontrollers. Microcontrollers zijn zoals het woord het aangeeft kleine controllers. Het deel *controllers* houdt in dat ze gebruikt worden om andere sensoren of systemen aan te sturen, zoals relays of LCD schermen. Dit doen ze met digital-to-analog of DAC en analog-to-digital convertors of ADC's. Die twee componenten onderscheiden microcontrollers van gewone computers. Een DAC wordt gebruikt om een computer analoge signalen te laten genereren om bv. een lampje hard of zacht te laten branden. ADC's gebruikt een microcontroller om sensoren met een variabel voltage zoals

bv. een ultrasone afstandssensor uit te lezen. Microcontroller's nemen dus de plaats tussen simpele ic's, die enkel gebruikt worden voor taken zoals timers of logische poorten, en computers, die gericht zijn op het verwerken van data en niet het aansturen van andere systemen. Bij een betaalterminal moeten er meerdere in en outputs (vanaf nu IO's) bestuurd worden, maar is het eigenlijke verwerken van data geen moeilijk taak. Een computer zou hier te krachtig zijn en het is daarmee ook te moeilijk om er bv. een lampje mee aan te sturen. Met simpele IC's zou de taak te ingewikkeld en moeilijk te programmeren worden.

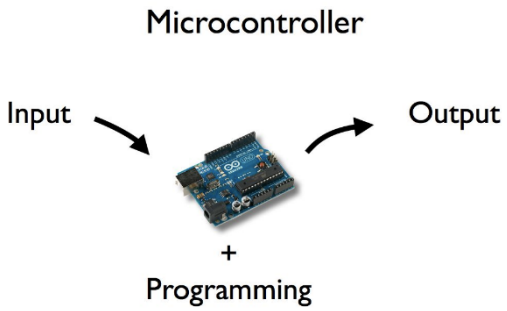


Fig. 2: De functie van een microcontroller.

Maar binnen de Microcontroller's is er nog een groot onderscheid te maken tussen system-on-a-chip systemen en simpelere oplossingen. SOC's hebben naast een CPU en RAM-geheugen vaak ook een videokaart of een NIC. De waarschijnlijk bekendste microcontroller in SOC-formaat is de Raspberry Pi. De Raspberry Pi heeft meerdere ADC's en DAC's, een NIC, videokaart en een - voor zo'n compact systeem - krachtige CPU. Dankzij die grote rekenkracht is het mogelijk om een volledig os zoals Raspbian OS - een speciale Linux distributie - te draaien op de Raspberry Pi. Daarnaast kan de processor van een Raspberry Pi ook multi-threaded workloads draaien. Het belangrijkste voorbeeld van een microcontroller dat geen volledig OS draait en bedoeld is voor single-threaded workloads is de Arduino. Op de Arduino programmeer je meestal in C++, of toch een soort dialect ervan, terwijl de Raspberry Pi alles van Python tot Ruby kan draaien. Een zelfs nog simpelere microcontroller is te vinden in bv. de ATtiny. Maar deze grenst al bij de familie van de geavanceerde IC's. Kleine microcontrollers zoals de ATtiny worden van IC's gescheiden door het feit dat ze programmeerbaar zijn.

Voor mijn toepassing, een digitaal betaalsysteem, is een microcontroller dus de oplossing. Genoeg IO's om schermpjes en bv. een NFC lezer aan te sturen maar niet te krachtig dat hij te duur wordt. Het eerste prototype dat ik wil bespreken is dat door ene Taweechai Maklay. Hij maakte een YouTube-video[7] die duidelijk alle aspecten weergeeft die zijn systeem te bieden heeft. Voor deze bespreking is zo'n demo genoeg aangezien ik me nog niet zal concentreren op de technische details zoals de code zelf.

Maklay maakt gebruik van een Raspberry Pi met wifi-verbinding. In zijn voorbeeld gebruikt hij als een feedback en output-systeem een webserver op de microcontroller zelf. In de demo zien we dat er naast de Raspberry Pi ook een NFC-lezer wordt gebruikt. Daarmee worden de gebruikers van dit systeem via een kaart zowel geïdentificeerd als geauthentiseerd. Na op de webinterface ingelogd te zijn met hun kaart kunnen gebruikers van het systeem geld afhalen of herladen en hun balans bekijken. Waar de data wordt opgeslagen wordt in de video niet getoond en een webconsole waar een gebruiker zich kan inloggen zonder kaart is ook niet te zien. Met dit systeem zien we wel dat NFC als fysieke identificatie en authenticatie manier en de Raspberry Pi als brein van een betaalterminal mogelijk zijn. Er is ook op te merken dat de betaalterminal maar een klein deel van de beschikbare rekenkracht nodig heeft. De webserver die op de Raspberry Pi wordt gedraaid is bv. niet echt nodig om gewoon te kunnen betalen in de blauwe refter, waar een simpel LCD scherm en keypad zouden volstaan. Zo'n webinterface zou wel interessant zijn op een plek waar, zoals in dit filmpje, al een computer aanwezig is, zoals op Scheppers bij het onthaal.

Een ander, meer uitgebreid prototype dat ik online[8] vond was dat van de YouTube gebruiker 'RzTV'. Ook hij maakt gebruik van een Raspberry Pi met NFC shield. Bij dit prototype wordt er geen webserver op de Raspberry Pi gedraaid, maar geeft het python programma feedback door een SSH console en door een LCD-scherm. De Raspberry Pi communiceert via een API met de webserver die op een computer lijkt te draaien. Dit systeem heeft daarnaast ook in tegenstelling tot het vorige een webconsole waar zowel leerlingen als administratieve personen zich kunnen inloggen. De console voor leerlingen heeft functies om de balans of een transactielog na

[7] (Maklay, 2014)
[8] (RzTV, 2016)

te kijken. De administratieve console heeft functies om bv. nieuwe NFC-tags te identificeren en om de balans van leerlingen aan te passen. Dit systeem is geschikter voor mijn toepassing aangezien ik leerlingen en ouders zeker de mogelijkheid wil bieden zich thuis in te loggen.

De laatste twee systemen die ik wil bekijken zijn wel een volledig systeem maar alleen voor een specifieke toepassing, namelijk van respectievelijk een koffiemachine en een automaat. Van dit eerste systeem is er een YouTube-video[9] door Inter Action. Ik koos dit specifieke systeem omdat het nog twee andere interessante aspecten demonstreert. De eerste zijnde het gebruik van bluetooth voor lokale configuratie van het systeem, zoiets zagen we in het eerste prototypesysteem van Maklay ook maar toen in de vorm van een webserver. Een derde optie zou uiteraard gewoon via een USB-verbinding zijn. Mijn ontwerp moet in elk geval een van deze oplossingen gebruiken voor bv. het registreren van nieuwe tags of het verwijderen ervan. Het tweede interessante punt van dit derde prototype is uiteraard dat het verbinding maakt met een koffieautomaat. Zo'n volledig geautomatiseerd systeem bij bv. een snoepautomaat moet zeker ook een mogelijkheid zijn met mijn systeem. Integratie van een microcontroller met een automaat is zeker mogelijk. Er zijn ook ouders die mijn enquête hebben ingevuld en vinden dat een automaat geen probleem zou moeten zijn voor een digitaal betaalsysteem: "Het zou handig zijn dat de leerlingen behalve een broodje ook een drankje zouden kunnen kopen. Met een elektronisch betaalsysteem is dit voor de school geen extra moeite vind ik!"

Een systeem met integratie met een automaat is dat in de Nottingham Hackspace. Een video[10] door Computerphile geeft een duidelijke uitleg over hoe dat systeem precies werkt. Er wordt gedemonstreerd hoe de automaat een publiek protocol genaamd MDB of multi-drop-bus gebruikt om met de microcontroller te communiceren. De microcontroller wordt dan bij een aankoop gevraagd om de aankoop te confirmeren. Het programma op de controller maakt een verbinding met een centrale server via ethernet om een aankoop te verifiëren.

[9] (Krohn, 2014)
[10] (Computerphile, 2014)

Deel B:
Ontwerp nieuw systeem

Met enerzijds de voor- en nadelen van andere systemen (deel A) en anderzijds de verwachtingen van de ouders (uit mijn enquête) schets ik in dit deel B het ideale betaalsysteem.

In deel C implementeer ik dat ontwerp dan in een volledig functioneel prototype. Maar een ideaal systeem kan je niet in één keer ontwerpen. Daarom gebruik ik dezelfde aanpak[11] als het softwarebedrijf *Flow Pilots* dat apps ontwikkelde voor onder andere Sodexo, AXA en BNP Paribas Fortis. Ik stel dus eerst de *doelstellingen* voor het systeem op, bekijk dan enkele *use cases* en maak tenslotte voor bepaalde elementen een *mock-up*. Omdat het systeem dat ik in deel C zal ontwikkelen een proof of concept is concentreer ik me alleen op essentiële functies: "If a feature doesn't help the business goals in any direct or indirect matter, it is deemed irrelevant" (Flow Pilots).

1. Doelstellingen

In dit hoofdstuk beslis ik wat het systeem moet kunnen om beter te zijn dan eender welk vorig systeem. Welke functies moet het zeker aan de gebruiker bieden, en waarop moet ik me dus concentreren?

1.1. Mogelijkheden

Het eerste doel is dat het systeem uitgebreide relevante mogelijkheden moet bieden. Dit geldt op veel vlakken. Voor de gebruikers betekent dit bijvoorbeeld dat als ze niet op school zijn ze toch informatie zoals hun balans moeten kunnen bekijken. Voor de betaalterminals betekent dit dat ze moeten kunnen worden ingezet in situaties waarbij er een leerkracht aanwezig is zoals bij de warme maaltijd maar bijvoorbeeld ook onbemand bij een drankenautomaat. Voor een familie op Scheppers betekent het dat het systeem alle mogelijke gezinsstructuren zoals bijvoorbeeld nieuw-samengestelde gezinnen moet ondersteunen. In het algemeen moet het systeem dus uitbreidbaar en dynamisch zijn.

1.2. Gebruiksgemak

Een tweede belangrijke doelstelling is dat het systeem gebruiksvriendelijk moet zijn. Het gaat hem voor een gebruiker niet om de betaling maar om hetgeen waarvoor men betaald. Gebruik van het systeem moet dus natuurlijk en ononderbroken gebeuren. Voor fysieke interactie met een betaalterminal betekent dit dus dat het simpel moet zijn om zich te identificeren. Bij een webconsole betekent dit dat een gebruiker informatie snel moet kunnen vinden. Als een gebruiker bijvoorbeeld zijn vorige transacties wil zien moet dat in zo min mogelijk stappen kunnen.

1.3. Veiligheid

Het derde doel is dat het systeem veilig moet zijn. Een aanvaller mag geen kans krijgen om gegevens te stelen of een betaling te doen in naam van een andere gebruiker. Daarom moet de communicatie tussen de betaalterminals en de website voldoende beveiligd zijn. Ook bijvoorbeeld de opslag van gebruikersgegevens moet zo gebeuren dat de aanvaller geen kans heeft om schade te berokkenen aan de school of een gebruiker.

[11] (FlowPilots)

2. Basiskenmerken

Vooraleer ik over de verschillende use cases (gebruikers scenario's) kan gaan moet er eerst over enkele basiskenmerken en -begrippen van het systeem beslist worden, zoals hoe een leerling of leerkracht zich kan identificeren? En welke soort terminals heeft het systeem nodig?

2.1. Fysieke identificatiemedia

Hoe een gebruiker zich kan identificeren en authenticeren met het systeem op het moment van een fysieke betaling zal het gebruiksgemak sterk beïnvloeden. Daarom is het belangrijk om voor een veilige maar ook snelle manier te kiezen.

Vanuit de vorige systemen en een literatuurstudie verzamelde ik enkele mogelijke methoden: via de studentenkaart, een sleutel met NFC of biometrische identificatie. Scheppers koopt studentenkaarten aan bij *Offini cards* en print daarop zelf de tekst. Daarom zou betalen met de studentenkaart ook via een bar- of QR-code kunnen. En dankzij Offini's grote aanbod zou betalen met de studentenkaart ook kunnen met bv. een magneetstrip. Dit is spijtig genoeg geen optie aangezien er niet veel microncontroller-compatibele magneetstriplezers voor consumenten bestaan. Daarbij zijn magneetstripkaarten vaak ook een stuk duurder dan NFC-kaarten. Een andere interessante optie is biometrische identificatie, het voordeel daarvan is dat het meteen ook als authenticatie fungeert, biometrische kenmerken kan men namelijk niet zo makkelijk vervalsen als een studentenkaart.

Maar ook biometrische identificatie is geen optie aangezien veel ouders dit niet OK vinden. Bijna 40% van de ouders die mijn enquête invulden gaven namelijk aan dat ze dit niet nodig vonden. Omdat biometrische identificatie en de privacy errond gevoelig ligt en zoiets toch maar zou kunnen ingevoerd worden voor de school indien alle ouders ermee akkoord zouden gaan zal ik deze identificatiemethode ook niet gebruiken.

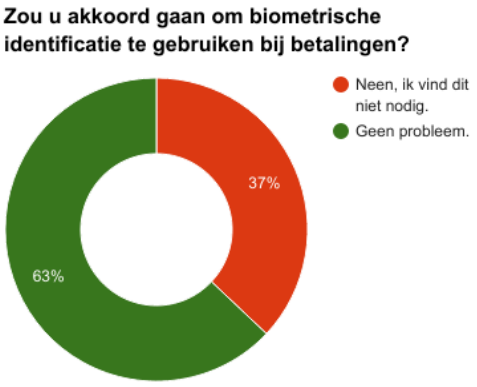


Fig. 3: Statistiek 1 enquête ouders.

Dan zijn de overblijvende mogelijkheden nog NFC, via studentenkaart of sleutel, of met een barcode op de studentenkaart. NFC en een QR-code verschillen op vlak van zowel het soort lezer, het produceren van de kaarten als de veiligheid van zo'n systeem. Het lezen van een barcode is zowel voor de gebruiker als voor het betaalsysteem al een moeilijkheid. De gebruiker zou zijn studentenkaart stil en op juiste afstand van de lezer moeten houden en de lezer zelf moet genoeg rekenkracht hebben om de afbeelding te verwerken. Dit sluit simpele microcontrollers voor het betaalsysteem zoals de Arduino al uit, met alleen SOC's zoals de Raspberry Pi als mogelijkheid. Een Raspberry Pi kost spijtig genoeg al snel €40 terwijl een Arduino met genoeg rekenkracht voor NFC maar €10 kost. Daarbij komt nog het verschil in compactheid tussen een camera en een NFC-reader. Een systeem met camera moet plaats voorzien om de barcode te kunnen lezen terwijl een NFC-antenne kan worden ingebouwd in bijna eender welk materiaal. De veiligheid van een barcode is ook een belangrijk aspect. Iedereen kan hem namelijk makkelijk door middel van een fotokopie dupliceren. De unieke id van NFC-tags kunnen ten eerste al niet makkelijk geëmuleerd worden, wat kopiëren of je eigen kaart aanmaken moeilijk maakt. Daarbij komt nog dat niet iedereen zomaar een NFC-kaarten printer heeft (maar daarentegen wel een inktjetprinter en lijm om barcodes na te maken). Het enige duidelijke voordeel van barcodes is dat ze goedkoper zijn dan NFC-tags. Op kleine schaal is dit misschien significant maar voor het printen van een massa kaarten is dit al minder een probleem. Om de veiligheid en gemak van het systeem te garanderen kies ik dan ook voor NFC als identificatiemethode.

De laatste beslissing die we moeten maken is om NFC ingebouwd in de studentenkaart of in een sleutel te gebruiken. In de enquête vonden 72% van de ouders betalen met de studentenkaart een goed idee, met een sleutel waren er dat maar 39%, bijna twee keer minder. De heer Vercauteren vertelde me in mijn interview wel dat een sleutel robuuster is. Maar omdat de mogelijkheid bestaat om studentenkaarten met NFC te printen en de studentenkaart überhaupt altijd op zak moet zijn bij de leerling vind ik dit een betere oplossing.

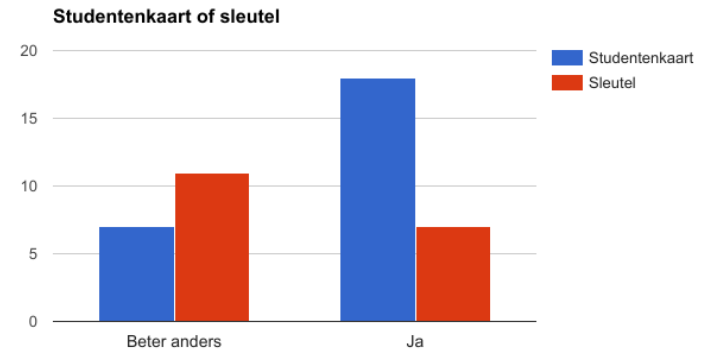


Fig. 4: Statistiek 2 enquête ouders.

2.2. Authenticatie

Het staat nu vast: mijn digitaal betaalsysteem zal gebruikmaken van de studentenkaart van de leerling met ingebouwde NFC-tag. Om nu voor zijn/haar maaltijd te kunnen betalen moet de gebruiker enkel de studentenkaart bij de betaalterminal te houden. Maar hoe weten we dat dit ook echt de studentenkaart van deze gebruiker is? Wat gebeurt er bij een gestolen kaart? Als extra beveiliging zou een leerling zich dus ook nog kunnen extra authenticeren met een pincode. Uit de enquête blijkt echter dat 74% van de ouders het voldoende veilig vinden om niet te authenticeren. Daarnaast zou de vlotheid van een transactie erdoor verminderen. En aangezien het bij dit systeem maar om relatief kleine bedragen gaat (€4.5 voor een warme maaltijd) en kaarten makkelijk kunnen worden geblokkeerd bij misbruik, lijkt extra authenticatie me onnodig.

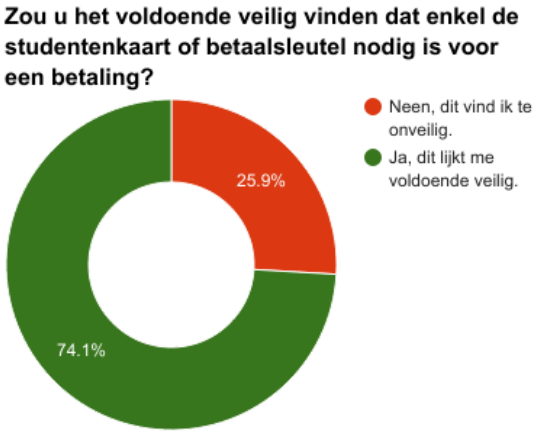


Fig. 5: Statistiek 3 enquête ouders.

2.3. Terminals

In de volgende hoofdstukken en in deel C zal ik het uiteraard vaak hebben over de betaalterminal. Maar niet elke betaling kan op dezelfde manier verlopen en er kan dus ook niet één soort terminal zijn. Daarom onderscheid ik drie type betaalterminals die elk andere functies hebben en op een andere manier met het systeem verbonden zijn.

De eerste terminal wordt gebruikt voor een normale betaling met een gebruiker en een operator. Dit is terminal type 1 of kort TT1. TT1 moet draadloos met het systeem verbonden zijn. WiFi is hiervoor een makkelijke optie aangezien het goedkoper is dan een eigen netwerk en er al een netwerk aanwezig is in de meeste gebouwen van Scheppers en bijvoorbeeld in de blauwe refter. Hier kan de terminal gebruikt worden voor warme maaltijden. Welke functies TT1 precies moet bieden zal ik in de verschillende use cases bespreken.

De tweede soort terminal is verbonden met een computer, waardoor die geen eigen wificonnectie nodig heeft. Met die verbinding geeft TT2 beheerders naast het accepteren van betalingen ook de mogelijkheid om NFC-tags

te beheren of om nieuwe kaarten te programmeren. TT2 zou ingezet kunnen worden bij het secretariaat van Scheppers.

Terminal type 3 tenslotte is een selfservice terminal waar alleen de gebruiker nodig is. Omdat er geen automaten op school gebruikt worden is terminal type 3 nu nog niet toepasbaar. Het ontwerpen ervan geeft wel meer mogelijkheden om mijn systeem in de toekomst verder uit te breiden. Hieronder een samenvattende tabel van de verschillende types terminals:

	Operator	Verbinding	Functies
TT1	Ja	WiFi	Betalen
TT2	Ja	PC	Betalen & beheer
TT3	Nee	WiFi	Betalen

Omdat het mechanisme voor een fysiek herlaadstation (om cash op de rekening te zetten) zeer geavanceerd is en niet essentieel voor de werking van het systeem heb ik besloten om deze niet in het ontwerp te nemen.

2.4. Webconsole

In de doelstelling 'Mogelijkheden' besprak ik al dat gebruikers de mogelijkheid moeten hebben om bijvoorbeeld hun rekeningbalans buiten school te kunnen opvragen. Hiervoor is er de keuze tussen een native app of een website. Een native app ontwerpen voor elk mogelijk platform (IOS, Android, Windows Phone…) vergt veel werk en vereist meestal ook een licentie om op hun respectievelijke appstore te publiceren. Een webconsole is een betere optie omdat een responsieve website compatibel is met bijna elk platform. Responsief betekend in webontwikkeling dat de website zich aanpast aan de grote van het scherm van de gebruiker. Wat die webconsole voor leerlingen, ouders en leerkrachten precies moet bieden bespreek ik in een use case over dat onderwerp. Voor beheerders zal er een andere webconsole zijn waar ze gebruikers en fysieke identificatiemedia (dus studentenkaarten met NFC tag) kunnen beheren. Via deze console kunnen ze ook met TT2 verbinden om nieuwe tags te registreren en bijvoorbeeld de gebruiker van een bepaalde tag op te vragen.

3. Use cases van het systeem

De simpelste manier om het perfecte systeem in te beelden is om vanuit het perspectief van de verschillende gebruikers ervan te kijken. Zo kunnen we makkelijk inzien wat het systeem voor elke gebruiker precies moet bieden. Hoe verloopt een interactie met het systeem? En hoe maken we die interactie voor de gebruiker zo vlot mogelijk?

3.1. Warme maaltijd op school (TT1)

Beginnende bij de meest essentiële functie van een digitaal betaalsysteem: het maken van een transactie. We bevinden ons in de blauwe refter, en een leerling of leerkracht wil een warme maaltijd betalen. Dit is dus een toepassing van het een terminal type 1, waarbij een operator de terminal bedient terwijl een leerling of leerkracht betaalt. We kijken eerst vanuit het perspectief van de leerling of leerkracht (of vanaf nu gebruiker genoemd).

3.1.1 Gebruiker

Wanneer een gebruiker aan de terminal komt zou men sowieso al direct zijn studentenkaart (of personeelskaart) moeten kunnen scannen. De gebruiker zou nooit moeten wachten tot de operator bijvoorbeeld een bedrag heeft ingetypt. Is het bedrag al ingetypt voordat de gebruiker zijn kaart scant dan moet dat bedrag zichtbaar zijn voor de gebruiker via bv. een schermpje. Als de gebruiker zijn kaart scant voordat de operator een bedrag heeft ingetypt dan moet het schermpje het huidige saldo en de naam van de gebruiker tonen. Als er dan zowel een

bedrag is ingetypt als een kaart is gescand geeft het schermpje alle informatie weer; de naam van de gebruiker, zijn vorige saldo, het te betalen bedrag en het saldo na betaling.

Er moet nu gecontroleerd worden of het saldo op de rekening van de gebruiker toereikend is. Bij die controle kan er uiteraard een probleem optreden als er bv. geen connectie is met de server. Een betaling zou dan wel nog *offline* moeten kunnen doorgaan. In zo'n geval zou de operator van de terminal bij elke betaling moeten beslissen of die doorgaat. Als er wel een verbinding met de server is kan het saldo van de rekening van de gebruiker toereikend zijn, enkele euro's ontoereikend, of al vóór de betaling onder nul. In die laatste twee gevallen moet de gebruiker dan toch nog zijn maaltijd kunnen betalen. Wanneer de rekening maar een paar euro's onder nul gaat zal de betaling sowieso doorgaan, met al dan niet een waarschuwend geluid of lampje. Als de rekening echter voor de betaling al onder de nul stond of te ver in het rood gaat moet de operator zelf kunnen beslissen of de betaling al dan niet doorgaat. Hoe die beslissing verloopt bespreek ik in de volgende use case.

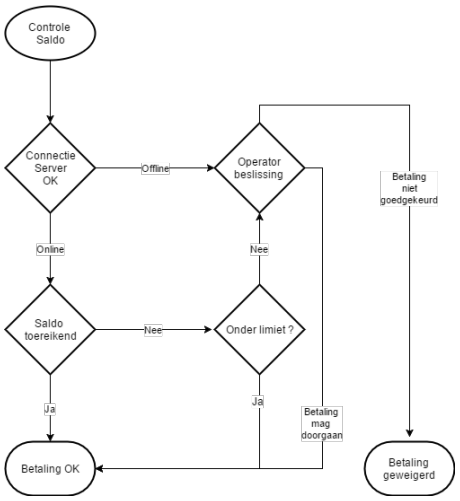


Fig. 6: Flowchart van een interactie door een operator met TT1.

De laatste stap voor het systeem is dan nog om feedback te geven of de betaling al dan niet goed is verlopen. Dit gebeurt sowieso via het schermpje en misschien ook met een extra geluidje of lampje. Een geweigerde betaling door ontoereikend saldo of een technisch probleem kan dan met een ander lampje en/of een andere toon aangegeven worden.

Nu we het hele betaalproces voor een gebruiker doorlopen hebben, kennen we al een deel van de functies die het systeem nodig heeft. Hieronder valt uiteraard al een microcontroller en NFC-lezer en daarnaast ook een display voor de leerling en optioneel een lampje of speaker. We weten ook dat er een limietwaarde voor aankopen met ontoereikend saldo moet kunnen worden ingesteld door een administrator van het systeem.

3.1.2 Operator

Voor een betaling in de blauwe refter is niet alleen de identificatie van een gebruiker nodig maar ook van iemand die de terminal bedient. Die operator moet ingeven welk bedrag de gebruiker moet betalen en in bepaalde gevallen beslissen of een betaling al dan niet doorgaat. We bekijken net zoals bij de vorige use case wat hij/zij zou doen bij een normale betaling.

Wanneer een gebruiker wil betalen moet het betaalsysteem eerst weten hoeveel er precies op de rekening van die gebruiker moet worden aangerekend. Een simpel *human input device* of HID is hiervoor een geschikte oplossing. Met zo'n HID kijkt de operator wat er precies betaald moet worden (bijvoorbeeld een dessertje en warme maaltijd) en voert die gegevens in op de terminal.

Nu het juiste bedrag is ingegeven moet het systeem nog weten wie de betaling maakt. Zoals eerder besproken gebeurt dat met de studentenkaart van de gebruiker en gaat de betaling door vanaf het moment dat die kaart wordt gescand door de NFC-reader. De verschillende opties zijn nu dus een succesvolle betaling, een technisch probleem of een betaling waarbij de operator moet beslissen of hij wordt geaccepteerd. Bij een technisch probleem zoals bv. een onderbreking van de connectie met de server moet een betaling nog steeds kunnen doorgaan. De microcontroller laat de operator dan beslissen over het verloop en slaat de ID van de studentenkaart en het bedrag dat wordt betaald lokaal op in een transactielog. Wanneer later de connectie met de server wordt hersteld kan dan die transactielog worden geüpload. Maar we gaan er uiteraard van uit dat de meeste betalingen met het systeem online kunnen gebeuren. Hierbij kan interactie door de operator zoals in de vorige use case besproken ook nodig zijn. Als het saldo op de gescande kaart na de betaling lager dan de ingestelde limietdrempel is moet de operator zelf kunnen beslissen of de transactie doorgaat. Hiervoor zou men

naast normale informatie zoals het huidige en nieuwe saldo van de gebruiker ook extra informatie zoals de laatste herlading van de rekening of de laatste transactie krijgen.

Dankzij deze use case weten we nu dat het systeem nog een extra display nodig heeft voor de operator.

3.2. Printergebruik op school (TT2)

3.2.1 Gebruiker

Betalen voor printergebruik zou voor de leerling praktisch op dezelfde manier gebeuren als voor een warme maaltijd. Het grootste verschil tussen deze twee transacties ligt in hoe de operator het systeem gebruikt.

3.2.2 Operator

Aangezien bij het secretariaat al een computer met toetsenbord beschikbaar is, is het keypad voor de operator nu overbodig. Hier is een verbinding met de computer voldoende om het systeem met input van de operator te voorzien. Ook een display voor de operator is onnodig aangezien een programma op de computer dezelfde functie kan invullen. Naast de microcontroller, NFC-reader en display voor de gebruiker is voor TT2 enkel een verbinding met de computer nodig, en dus geen extra display. Andere functies besproken in 'Bediening van de betaalterminal 1: warme maaltijd' zoals het invoeren van het juiste bedrag of beslissen of een transactie al dan niet doorgaat bij een te laag saldo worden dus op de computer verwerkt. De technische details van de verbinding tussen microcontroller en computer zal ik in deel C: *Implementatie* bespreken.

3.3. Gebruik automaat op school (TT3)

3.3.1 Gebruiker

Bij terminal type 3 is er geen operator nodig om het juiste bedrag in te typen, de terminal is namelijk bedoeld om enkel door de gebruiker te worden bediend. Ik kan geen scenario bedenken waarin Scheppers zo'n type terminal zou kunnen implementeren op school, aangezien de school geen drank- of snack-automaten meer aanbiedt. Maar ik bespreek de werking ervan wel, voor moest dat in de toekomst wel zo zijn.

Hoe een gebruiker omgaat met de terminal is praktisch hetzelfde als bij TT1 of TT2. De microcontroller wacht namelijk tot er ofwel een kaart van een gebruiker wordt gescand of dat er een bedrag wordt bepaald door in dit geval niet de operator maar bv. de drankautomaat. In het laatste voorbeeld dat ik gaf van alternatieve systemen (A 3), besprak ik al kort hoe een drankautomaat communiceert met betaalsystemen. Wanneer op de automaat een selectie wordt gemaakt communiceert de controller van de drankautomaat via een speciaal netwerk het te betalen bedrag. Elk van die betaalsystemen kan dan reageren op dat signaal. Dit geeft dus de mogelijkheid om oftewel met muntjes of met een digitaal systeem te betalen. In ons geval reageert de microcontroller alleen op het betaal-signaal wanneer er een kaart wordt gescand. In tegendeel tot bij TT1 kan een gebruiker niet verder dan een bepaalde waarde onder nul gaan, omdat er geen controle op is en er dus potentieel misbruik zou kunnen worden gemaakt. Ook offline betalingen moeten kunnen worden uitgeschakeld om datzelfde probleem te vermijden.

3.4. Webconsole gebruiker

In deze use case bespreek ik hoe een gebruiker zijn resterende saldo zou nakijken als hij niet bij een terminal is. Zoals onder *basiskenmerken* is beslist moet een gebruiker hiervoor een webconsole ter beschikking hebben.

Een webconsole voor de leerling zou een manier moeten bieden om de rekening op te laden, een zicht moeten geven op vorige transacties en op de huidige balans. Een snel menu en een propere userinterface moeten deze interactie vlot laten verlopen. Men moet zijn rekening kunnen herladen en deze functie moet met de meest gebruikte manieren voor online betaling (mastercard, maestro, visa, paypal…) compatibel zijn. Bij het nakijken

van transacties moet de leerling kunnen zien van welke terminal een transactie is gemaakt, over wat voor betaling het gaat (warme maaltijd, printer, automaat…), de datum van de transactie en uiteraard het bedrag. De pagina met de huidige balans moet een geschiedenis geven van vorige saldo's. En als optionele functie in de toekomst zou er ook een voorspelling kunnen gegeven worden van wanneer de balans op nul zou komen te staan op basis van de betalingshistoriek. Op een instellingenpagina moet een gebruiker zijn wachtwoord, herstel e-mailadres en fysieke media kunnen beheren. Bij diefstal moet men die fysieke media hier ook kunnen blokkeren.

Speciale gevallen zoals een familie met twee kinderen die dezelfde rekening en dus transactielog delen of een oudere leerling die administratieve machtiging heeft over een jongere broer of zus zal ik nog later onder *Ontwerp* bespreken.

3.5. Ouder gebruikt webconsole

Ook ouders moeten zich thuis of mobiel kunnen inloggen om de rekening van hun zoon of dochter te beheren. Elke ouder heeft voor de webconsole zijn of haar eigen login-gegevens, om tegemoet te komen aan bijvoorbeeld nieuw-samengestelde gezinnen of andere familiale situaties. Speciale toegangscontrole moet dus per gebruiker beheerd worden en niet via een simpele 'ouder' of 'leerling' rol. Hoe dit in z'n werk ga bespreek ik later in meer detail. In een standaard geval verschilt de webconsole van een ouder niet veel van die van een leerling. Onder bepaalde gebruikersrechten kan de ouder ook andere gebruikers-accounts beheren en rechten veranderen.

3.6. Webconsole administratie

Wanneer de administratie van de school (bv. het secretariaat) het betaalsysteem wil beheren moeten ze daarvoor een eigen webconsole hebben. Die console is anders dan die voor de gewone gebruiker omdat de administratie uiteraard alle gebruikers moet kunnen beheren, terwijl een gewone gebruiker enkel toegang mag hebben tot zijn eigen gegevens. De console heeft dus een ander ontwerp dan die voor gebruikers, die het makkelijk maakt om bv. de databank met gebruikers aan te passen, transacties en de balans aan te passen en fysieke media te beheren. Die fysieke media moet men bij diefstal of verlies kunnen blokkeren of als dat nodig is terug activeren. Wanneer de console wordt geopend op een computer die met een betaalterminal is verbonden (zoals bij 'Printergebruik op school': 'bediening betaalterminal') moet er de mogelijkheid zijn om nieuwe NFC-tags te registreren of informatie over een bepaalde tag op te vragen.

4. Ontwerp

Nu ik het systeem vanuit alle mogelijke perspectieven van de gebruiker en administratie heb bekeken, bespreek ik samengevat welke functies nodig zijn op vlak van hard- en software om aan alle vereisten te voldoen. Het toepassen van die functies bespreek ik in deel C: *Implementatie*.

4.1. Identificatiemedia

Zoals in het hoofdstuk over basiskenmerken al besproken is zal het systeem NFC-kaarten van Offini cards gebruiken om gebruikers te identificeren met betaalterminals. Precies welke kaarten daarvoor geschikt zijn en hun technische details bekijk ik in deel C. In de toekomst zou het systeem wel andere vormen van fysieke identificatiemedia moeten ondersteunen. Bijvoorbeeld als identificatie via QR-code toch voordelig blijkt te zijn.

4.2. Front-end hardware

De front-end hardware zijn de componenten waarmee een gebruiker, de administratie of een operator zullen interageren. De verschillende type terminals verschillen onder meer in welke componenten ze bevatten. Hieronder een schema voor een van de drie type terminals: TT1. Het schema is opgedeeld in de zijde van de gebruiker in het paars en die van de operator in het oranje. De kleur en uiteinden van de pijlen geven het type verbinding weer die ik onder 'Communicatie' bespreek.

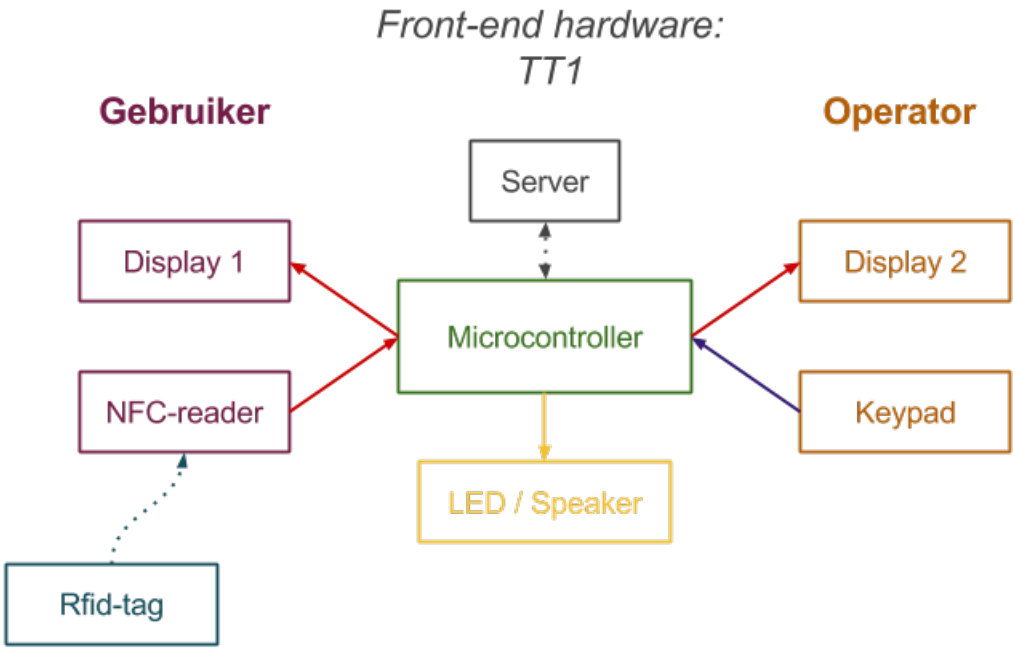


Fig. 7: Schema front-end hardware voor TT1.

4.2.1 Microcontroller

De verschillende type terminals hebben andere vereisten op het vlak van microcontrollers. Zo hebben terminal TT1 en TT3 een eigen internetverbinding nodig maar TT2, die met een computer (op het secretariaat) verbonden is, niet. Het betaalsysteem heeft dus twee verschillende soorten microcontrollers nodig waarrond de rest van de terminal is opgebouwd. Zowel het offline- als online type microcontroller heeft genoeg IO nodig om de displays, de NFC-reader en optioneel LED's of een speaker aan te sturen.

4.2.2 Input

Gegevens komen de microcontroller binnen via drie verschillende wegen. De eerste zijnde het HID van de operator. Bij TT1 is die direct verbonden met de microcontroller, bij TT2 via de computer en bij TT3 is er geen input van de operator nodig aangezien er ook geen operator is. De tweede input komt via de NFC-reader van de NFC kaart van de gebruiker. Er moet dan dus ook beslist worden welk soort reader het systeem zal gebruiken. De laatste manier van input is door middel van een internetverbinding met de (app)server. Via die verbinding moet de terminal bijvoorbeeld de huidige balans of naam van een gebruiker kunnen opvragen.

4.2.3 Output

Feedback aan de gebruiker en de operator gebeurd vooral door displays. Terminal type 1 heeft twee displays nodig, één voor de gebruiker en één voor de operator. TT2 en 3 hebben maar één display nodig voor de gebruiker. De displays voor gebruiker en operator moeten genoeg ruimte hebben om alle nodige informatie weer te geven en moeten duidelijk leesbaar zijn. Ze zouden best ook niet te veel IO's innemen op de

microcontroller en dus een bus-protocol zoals I2C of SPI gebruiken. Een optionele functie is ondersteuning voor LED's of een speaker om extra feedback te geven of een betaling al dan niet succesvol was.

Output aan de operator bij TT2 verloopt ook via de verbinding met de computer. Welke informatie wordt weergegeven blijft hetzelfde, alleen wordt die nu door een programma op de computer weergegeven.

Feedback aan het centrale systeem verloopt over dezelfde draadloze verbinding als die waarover de terminal zelf informatie stuurt. Over die verbinding stuurt de terminal bijvoorbeeld het betaalde bedrag en een beschrijving van wat er precies gekocht werd.

4.2.4 Communicatie

Rode pijlen geven aan dat communicatie met de NFC-reader en de twee displays via een seriële bus-interface zal gaan. Dit moet wel aangezien de microcontroller anders niet genoeg IO over zou hebben. De donkerblauwe pijlen duiden een simpele analoge of digitale connectie aan, namelijk voor het HID (het keypad) en LED's of een speaker. De gestippelde pijlen wijzen op een draadloze connectie: via NFC met de studentenkaart en via WiFi met de appserver. Bij TT2 is die draadloze verbinding niet nodig en wordt die vervangen door een verbinding met een computer.

4.3. Back-end hardware

De back-end hardware bestaat uit een web-, app- en database-server. De webserver serveert statische bestanden voor de consoles. De appserver draait het programma dat alle beslissingen over betalingen en gebruikers neemt. De databaseserver communiceert met de appserver en slaat alle informatie van transacties tot gebruikers op. Wat de software op deze servers moet kunnen bespreek ik onder *Software* en later tijdens de implementatie.

4.4. Front-end software

Dit is de interface die de gebruiker te zien zal krijgen, via bijvoorbeeld de webconsole of het display van een betaalterminal.

4.4.1 Arduino

Dit is de software die zal draaien op de microcontroller die in de betaalterminal wordt gebruikt. In dit hoofdstuk geef ik zelf nog geen code of een flowchart van de software, enkel een lijst van functies.

Terminal T1

De software op de terminals heeft als belangrijkste taak om gegevens te verzamelen en die dan met de appserver uit te wisselen. Daarom moet het programma functies hebben voor het uitlezen van het HID, het opslaan van gegevens over een gescande NFC-tag en het communiceren met de server. De belangrijkste lus in het programma moet continu die functies uitvoeren. Wanneer alle benodigde gegevens zijn doorgestuurd moet de server terugzenden of de transactie succesvol was of dat het saldo van de gebruiker ontoereikend is. Hieruit leidt de terminal af hoe feedback moet geven worden aan de gebruiker en de operator. Als het saldo ontoereikend is of het systeem offline is dan moet de operator kunnen beslissen om de transactie al dan niet toch te laten doorgaan. In dat laatste geval, dus wanneer de transactie offline doorgaat, moeten de gegevens lokaal worden opgeslagen. Als het systeem niet bezig is met gegevens verzamelen en offline is moet er gecheckt worden of er opnieuw een connectie met de server gemaakt kan worden. Als het systeem online is moet de software in een soort idle-cyclus belanden, waar het blijft pollen voor input en om de paar minuten een *heartbeat* naar de server stuurt om de operator op tijd te kunnen waarschuwen dat het systeem offline gaat.

Dit zijn alle benodigde functies van terminal type 1. In deel C van het eindwerk zal ik bespreken hoe deze functies in praktijk zouden worden geprogrammeerd.

Terminal T2

In dit tweede hoofdstuk bespreek ik de benodigde functies van terminal type 2, welke verbonden is met een computer. Deze terminal wordt zoals eerder vermeld naast normale betalingen ook gebruikt voor het beheer van NFC-tags.

De betaalactiviteiten van deze terminal verlopen grotendeels zoals die van TT1. Om de software over het hele betaalsysteem consequent te houden is het nog steeds de microcontroller - en niet de computer - die het rekenwerk op zich neemt. Type 2 verschilt echter op het vlak van input, feedback aan de operator en de connectie met de server. Input met het keypad wordt niet meer gedaan door het keypad te pollen maar door na te kijken of de seriële buffer van/naar de computer informatie bevat. Het computerprogramma zendt meteen nieuwe informatie naar de microcontroller als er toetsen worden ingedrukt. Feedback aan de operator over bijvoorbeeld de gebruiker wordt door de microcontroller naar de computer gestuurd. De laatste functie van de betaalactiviteit die verschilt van TT1 is die om gegevens naar de server te sturen. Deze functie wordt ingevuld door een systeem dat gegevens over de computerverbinding stuurt. Wanneer de server dan antwoordt op een aanvraag stuurt de computer die gegevens door naar de microcontroller. De andere betaalfuncties zoals feedback geven aan de gebruiker of NFC-tags detecteren verlopen op exact dezelfde manier als bij terminal type 1.

Een totaal andere set functies van terminal type 2 tegenover TT1 is die voor de beheersactiviteiten. Deze functies dienen om nieuwe tags te registreren of informatie over een tag op te zoeken. Dit systeem werkt als een extensie van de beheerders-webconsole. Het programma van die webconsole communiceert met een programma dat op de computer draait. Dat programma verbindt op zijn beurt via de seriële interface met de microcontroller en kan zo NFC-tags laten beschrijven. Hoe precies dat lokale programma en webconsole werken bespreek ik in hun respectievelijke hoofdstukken.

4.5. Webconsole gebruiker

De webconsole is een directe manier waarmee de gebruiker met het systeem in contact komt. Hoe de console eruitziet en werkt moet dan ook goed uitgedacht zijn. Via de console moeten leerlingen, ouders en leerkrachten informatie kunnen ophalen zoals hun huidige balans, een transactielog of informatie over identificatiemedia. Daarnaast moeten ze instellingen kunnen wijzigen in verband met hun account en dat van hun zoon of dochter in het geval van een ouder. Ik bekijk eerst het uitzicht van de console en daarna wat de software die op de achtergrond draait moet kunnen. Allebei deze onderwerpen worden verder uitgewerkt in deel C.

4.5.1 Ontwerp

De webconsole moet responsief en dynamisch zijn. Responsief betekent dat de website ook op bijvoorbeeld een smartphone of een tablet moet kunnen worden gebruikt. Dynamisch betekent dat de informatie op de website veranderd maar ook niets anders. Er moet niet telkens een nieuwe pagina geladen worden wanneer een gebruiker nieuwe informatie opvraagt. De menubalk bv. moet doorheen de navigatie van de webconsole niet veranderen. Met dat in gedachte ontwierp ik wireframes van de verschillende pagina's van de console. De eerste zijnde de thuispagina die de belangrijkste informatie moet weergeven: de huidige balans op een rekening.

Om de pagina responsief te maken ontwierp ik de thuispagina met panelen om informatie weer te geven. Die panelen kunnen dan onder elkaar staan als de pagina op een kleiner scherm wordt bekeken. Een leerling zou hier maar één rekening zien terwijl ouders er een zouden zien voor elk van hun zonen of dochters. Via de menubalk kan men naar andere pagina's navigeren en in de rechterbovenhoek vindt de gebruiker informatie over wie er is ingelogd. Daar kan men zich dan ook uitloggen.

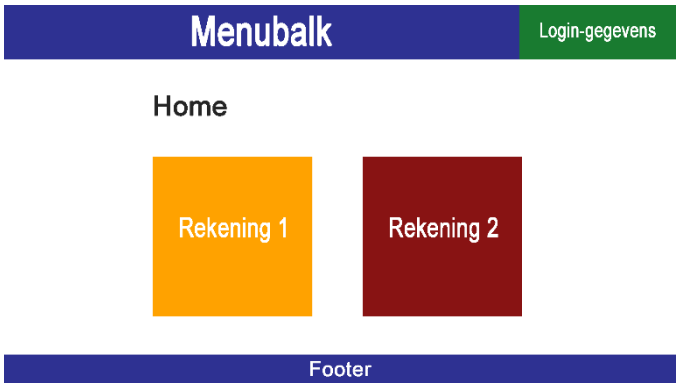


Fig. 8: Wireframe 1 van de webconsole voor gebruikers.

Een andere belangrijke functie die de webconsole moet vervullen is het nakijken van transacties. Zo moet een ouder bijvoorbeeld kunnen nakijken wat zijn of haar zoon of dochter recent aankocht. De transactiepagina moet dus informatie zoals de tijd van een transactie, het bedrag dat is overgezet en een algemene beschrijving weergeven. Als er meerdere rekeningen onder een gebruikers account vallen moet het mogelijk zijn om alleen de transacties op een bepaalde rekening te bekijken. Heel simpel voorgesteld zou die pagina er zo kunnen uitzien:

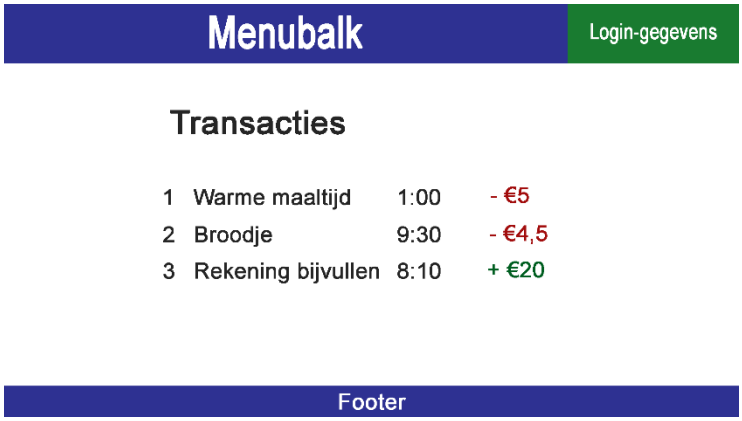


Fig. 9: Wireframe 2 van de webconsole voor gebruikers.

Het is belangrijk om op te merken dat enkel de informatie in het midden van de pagina veranderd. De menubalk en login-gegevens bovenaan en extra informatie in de footer onderaan moeten niet herladen worden. De transactiepagina moet ook de mogelijkheid bieden om een rekening te herladen.

De derde en laatst pagina is die met de instellingen van de gebruiker. Hier kan men dan zijn wachtwoord veranderen, de rechten van een zoon of dochter, fysieke identificatiemedia en later misschien de verbinding met externe systemen beheren.

Software

De taak van internetbrowsers is niet alleen om pagina's te renderen maar ook om op de achtergrond software te draaien. Die extra software die in de browser van gebruikers draait laat men op een website zoeken en maakt dynamische websites mogelijk.

Dat laatste is wat de webconsole vooral zal gebruiken. Eén van de functies is namelijk om nieuwe pagina's op de achtergrond op te vragen. Die functie moet geactiveerd worden wanneer er op een link in de menubalk wordt gedrukt. Een andere functie moet data opvragen van de server. Aan de hand van de pagina waarop een gebruiker zit vraagt deze functie om de paar minuten nieuwe gegevens aan. Uiteraard kan een gebruiker de functie ook zelf activeren als men over de nieuwste informatie wilt beschikken. De software moet ook informatie naar de server doorsturen als er bijvoorbeeld in de instellingen een wijziging wordt gemaakt.

Webconsole administratie

De webconsole voor de administratie is voor mijn prototype van een lagere prioriteit om de potentiele functionaliteit van het systeem te demonstreren. Aangezien die console ook niet op snelheid en dynamiek gericht zou zijn bespreek ik hem dan ook in veel minder detail dan de gebruikersconsole.

Uiteindelijk zou de administratie console alle informatie over gebruikers, hun balans, hun rechten, rekeningen en fysieke media moeten kunnen aanpassen. Of die functies nu in een mooi jasje gehuld zijn doet niet ter zake, als ze maar vlot toegankelijk zijn. Voor een simpel ontwerp dacht ik aan verschillende panelen voor transacties, gebruikers, rekening… Boven elke kolom zou men moeten kunnen zoeken om makkelijk een bepaalde gebruiker of rekening te vinden. De informatie voor dat object zou dan getoond worden met koppelingen naar gerelateerde objecten. Als men bv. in de tabel met transacties naar een bepaalde datum zoekt zou men op de accountnaam van die transactie kunnen klikken om de instellingen ervan te bekijken. Wanneer men de tabel met fysieke media doorzoekt kan men ook een nfc-tag scannen als men met TT2 verbonden is. Als men dan gezocht heeft in een bepaald paneel, bv. die met de rekeningen kan men daar ook informatie zoals de balans aanpassen.

4.6. Local proxy administratie

De local proxy voor de administratieve console is een klein programma dat op een computer draait en drie functies voldoet. Ten eerste zorgt het voor de internetverbinding van TT2, ten tweede geeft het feedback aan de operator en ten derde fungeert het als een verbinding tussen de terminal en de administratieve webconsole. De functies van dit stukje software zijn dus vooral aan de ene kant de verbinding met de terminal en aan de andere kant die met de webconsole in de gaten houden. Als één van de twee een bericht wilt sturen naar respectievelijk het internet en de terminal is de taak van de proxy om dat bericht naar hun bestemming door te sturen. Een extra functie is om informatie voor de operator van de terminal weer te geven. Die informatie is dezelfde als bij het display van terminal type 1, alleen wordt die nu via een computer aangegeven.

4.7. Back-end software

Dit is de software die niet de gebruiker, noch een beheerder ooit te zien zal krijgen. Het vervult nochtans de belangrijkste functies en is het hart van het betaalsysteem. Het totale back-end systeem is in enkele delen verdeeld, die delen vormen de zogenoemde *server stack*. Ten eerste is er de server die statische bestanden voor de webconsoles aan de webbrowser van de gebruikers zal serveren. Dit noemen we de webserver. Een tweede server is de database-server. Hierop wordt alle mogelijke data van gebruikers tot transacties op opgeslagen. De allerbelangrijkste server is de appserver. De software hierop reageert op aanvragen van alle kanten zoals de terminals en de webconsole. Nog een ander deel van het backend systeem is de betaal-API, die dient om betalingen van gebruikers - om hun rekening aan te vullen - aan te rekenen.

Hieronder volgt een simpel schema van de complete server stack in het midden en de twee front-ends, fysiek en in virtueel, resp. links en rechts. Hoe de verbindingen precies werken bespreek ik naast een detail van elke server in deel C.

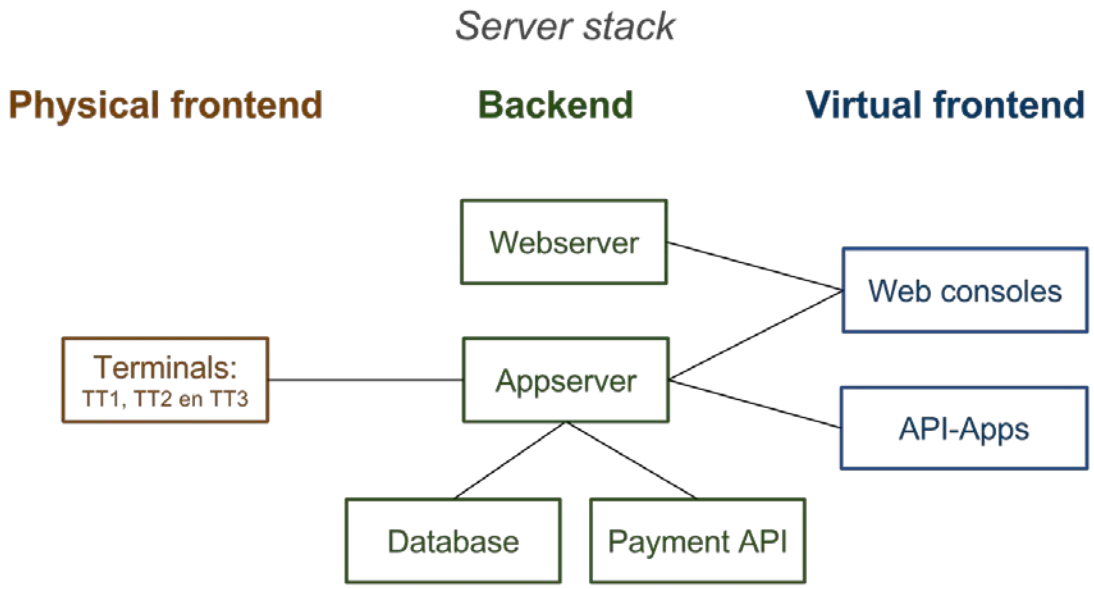


Fig. 10: Schema van de server stack.

4.7.1 Webserver

Ik bekijk eerst de simpelste van de servers. De enige taak van de webserver is om te reageren op aanvragen van clients (zoals de webbrowsers van gebruikers) en het juiste bestand terug te sturen. Wanneer een client bijvoorbeeld een bepaalde pagina wilt renderen of afbeelding weergeven stuurt de webserver die resource terug. Het gaat hier dus alleen om statische bestanden, die niet afhangen van wie er is ingelogd. Dit onderscheid tussen persoonlijke data en statische resources maakt het beheren van de servers een stuk simpeler. Het programmeren van deze server bestaat hem enkel in enkele lijnen code in een configuratiebestand. Zo weet de server hoe die op bepaalde aanvragen moet reageren.

4.7.2 Databaseserver

Ook de databaseserver vergt relatief weinig configuratie. Er moet wel nagedacht worden over de database-structuur. Deze structuur zal uiteindelijk ook bepalen hoe de appserver werkt. Voor mijn toepassing - veel, gestructureerde data - is een zogenoemde *relational database* ideaal. Relational databases werken met rijen en kolommen, waar een cel een relatie aangeeft tussen een rij en een kolom. Zo'n structuur is te vergelijken met het Microsoft-office programma Excel, waarbij men een blad vaak structuur geeft met namen in de bovenste rij. Deze kolomnamen noemt men *keys* in relational databases. Er bestaan uiteraard andere database types voor andere doeleinden maar die komen in dit eindwerk verder niet ter sprake.

Relational databases zijn opgesplitst in *tabellen*. Tabellen, zoals sheets naar analogie met Excel, zijn de eigenlijke plek waar je data opslaat. Zo kun je meerdere tabellen in één database hebben. Velden in een tabel kunnen naar elkaar of naar een andere tabel verwijzen. De key van een kolom met velden die naar een andere tabel verwijzen noemen we een *Foreign key*. Zo kan een gebruiker bijvoorbeeld een verwijzing hebben naar zijn rekening. Hoe de tabellen in een database zijn opgebouwd en welke foreign keys naar waar verwijzen beïnvloeden de snelheid van het programma dat de database zal gebruiken.

Ik verdeel de database van het betaalsysteem in vier tabellen.

Users

De eerste tabel is die met de gebruikers. Elke gebruiker, dus elke leerling, leerkracht, ouder, of zelfs een administratief persoon zit in deze tabel. Merk op dat administratieve gebruikers twee accounts kunnen hebben;

één om te betalen en één om het systeem te beheren. Deze tabel noemen we vanaf nu de *Users* tabel. De user tabel bevat een user-number, user-ID, gebruikersnaam, e-mailadres, persoonlijke instellingen, rekenings-, beveiligings-, betalings-, relatie- en toegangsgegevens.

User#	User-ID	Name	E-mail	Settings	Security	Payment	Relationship	Access

De user# (-number) is bedoeld om bijvoorbeeld van één tabel naar een andere te verwijzen met foreign keys. Dit maakt de user# de *primary key* van de tabel. De user-ID is een unieke manier om gebruikers te identificeren zonder gebruik te moeten maken van een gebruikersnaam die in de toekomst zou kunnen veranderen. Het verschil tussen de user# en -ID is dat de eerste via een incrementeel telsysteem wordt gegenereerd, de tweede ad random voor elke nieuwe gebruiker. Omdat men met een incrementeel telsysteem gewoon de user# van een gebruiker kan raden en mogelijks exploiteren mag deze enkel intern gebruikt worden, binnen de software. De User-ID daarentegen kan men niet zo makkelijk raden en kan daarom gebruikt worden tijdens bijvoorbeeld externe communicatie (tussen gebruiker en server). Het e-mailadres kan gebruikt worden voor verschillende doeleinden van wachtwoordherstel tot later als optionele functie voor notificaties bij bijvoorbeeld een nieuwe betaling. Persoonlijke instellingen zijn niet essentieel maar kunnen later bijvoorbeeld de taalvoorkeur van de gebruiker bevatten. Rekeningsgegevens bevat een lijst met rekeningen die onder de gebruiker vallen. Zo kan een ouder bijvoorbeeld een rekening hebben voor elk kind. Beveiligingsgegevens bevatten alle nodige gegevens om een gebruiker zijn user-account veilig te stellen. De betalingsgegevens bevatten informatie over eerder gebruikte kaarten bij de betalings-API om betalingen vlotter te doen lopen en maandelijkse betalingen mogelijk te maken. Relatiegegevens geven aan met welke andere gebruikers deze user verbonden is, zoals een leerling met een ouder. Toegangsgegevens bepalen welke gebruiker welke acties kan ondernemen met het account. Bij een leerling heeft een ouder bijvoorbeeld alle toegang terwijl een broer of zus maar enkele instellingen kan wijzigen.

Accounts

De tweede tabel is die met de rekeningen, of vanaf nu *Accounts*. Deze naam is niet te verwarren met een useraccount, welke gewoon een gebruiker is. Het gaat hier over een soort van spaarpot die door meerdere gebruikers beheerd kan worden. Zo kan een familie bijvoorbeeld één rekening hebben voor alle zonen en dochters of gewoon voor elk kind één.

#	UID	Name	Balance	Access

De tabel accounts heeft net zoals users een account-nummer en -ID en toegangsgegevens met dezelfde functie. Daarnaast heeft het een key voor de naam en de balans.

Media

De derde tabel, *Media*, bevat de fysieke identificatiemedia. Deze tabel is gescheiden van de account- en usertabel om meer mogelijkheden te bieden voor de toekomst. Zo kan een familie bijvoorbeeld een gemeenschappelijke rekening gebruiker maar aparte NFC-kaarten hebben voor elke leerling.

#	UID	Name	Account	NFC-ID	Secret	Access

De tabel met fysieke identificatiemedia heeft ook een media-nummer en -ID en toegangsgegevens voor elke kaart of QR-code. Het heeft ook een veld voor een naam, rekening, NFC-ID en een secret. De naam wordt bij een betaling getoond. Dit kan dus de naam van een gebruiker zijn zoals ‘Pieter Fiers’ of gewoon een familienaam. Het veld met de rekening is een foreign key die naar de Accounts tabel verwijst. De rekening waarnaar die verwijst is degene die wordt aangerekend bij een betaling. De NFC-ID is een voorgeprogrammeerde code op elke NFC-tag, deze code is uniek voor elke tag, zoals bij een MAC-adres. Dit geeft de optie om de NFC-tags van studenten tegen kopiëren te beveiligen. Maar net zoals MAC-adressen gespoofd kunnen worden kunnen ook NFC-ID’s geëmuleerd worden. Al lijkt het me dus niet nodig om de ID te gebruiken, sla ik hem toch op. Een secret is een code die extern gebruikt kan worden maar die niet bedoeld is voor de gebruiker en dus geheim is. Ze dient

voor de authenticatie van de NFC-tag en moet op de tag geschreven worden wanneer die met het systeem wordt geregistreerd.

Transactions

De vierde tabel is de *Transactions* tabel. In deze tabel staan uiteraard alle transacties.

#	UID	Description	Service	Time	Media-ID	User-ID	Account-ID

Net zoals met alle vorige tabellen heeft deze een transaction-# en ID. In tegendeel tot bij de vorige drie tabellen heeft een transactie geen toegangsgegevens aangezien er aan doorgevoerde transacties niets meer te veranderen valt. De tabel heeft verder een bedrag, beschrijving, servicenaam, tijd, media-ID, user-ID en rekening-ID. Het *bedrag* is het betaalde bedrag. De *beschrijving* geeft kort aan waar de transactie over gaat bijvoorbeeld 'Warme maaltijd' of 'Maandelijkse herlading'. *Servicenaam* geeft aan via welke methode de betaling is gemaakt, zoals 'TT1: Blauwe refter' of 'Online betaling'. De tijd geeft de tijd van transactie aan, beslist door de server, dus niet wanneer een kaart wordt gescand maar wanneer die transactie ook echt wordt aangerekend. Media-ID en user-ID zijn foreign keys en wederzijdse exclusief, er kan dus maar een van de twee ingevuld zijn. Media-ID verwijst naar een fysieke identificatiemedia waarmee de betaling werd doorgevoerd uit de media tabel. User-ID is ingevuld wanneer een gebruiker zelf de transactie uitvoerde, oftewel in hun naam via bijvoorbeeld een API. Rekening-ID is een foreign key uit de account tabel. Deze bepaald aan welke rekening de betaling werd aangerekend.

Admin

Omdat beheerders bijvoorbeeld geen e-mailadres of persoonlijke instellingen nodig hebben zoals een normale gebruiker is een gescheiden tabel handig. In die tabel moet dus alleen het essentiële zoals een naam en beveiligingsinformatie staan.

Sessions

Een laatste tabel is de *Sessions* tabel. Hierin worden sessies van gebruikers op de webconsole in opgeslagen. Zo blijft men ingelogd en moet men zich niet steeds authenticeren telkens men de webconsole vanaf een andere webbrowser bekijkt. Welke keys in deze tabel staan hangt vooral af aan de zogenoemde session manager die ik zal gebruiken. Dit is een stukje software dat zoals de naam al aangeeft gebruikerssessies opslaat en nakijkt.

Samengevat zien al die tabellen er dus zo uit:

Tabellen	***Columns***								
Users	#	UID	Name	E-mail	Settings	Security	Payment	Relationship	Access
Accounts	#	UID	Name	Balance					Access
Media	#	UID	Name	Account	NFC-ID	Secret			Access
Transactions	#	UID	Description	Service	Time	Media-ID	User#	Account-ID	
Admin	#	UID	Name	Security					
Sessions	?								

4.7.3 Appserver

De appserver beslist over de eigenlijke betaling en is afhankelijk van de databaseserver voor de gegevens om betalingen door te voeren. Zoals visueel geïllustreerd in het begin van dit hoofdstuk *Back-end software*, moet de appserver reageren op aanvragen van de webconsoles, terminals en andere applicaties.

Er kan informatie opgevraagd en aangepast worden in verband met al de verschillende aspecten van het systeem zoals gebruikers en transacties. Om die aanvragen te organiseren moeten we de appserver voor elke soort informatie in zogenoemde paden of *paths* verdelen. Voor mijn toepassing zijn de nodige paths '/data, '/auth' en '/app'. Het data-path dient om applicaties een interface te geven aan de gegevens van het betaalsysteem. Via

deze interface kunnen die applicaties alle mogelijke objecten uitlezen of aanpassen. Van een ‘user’ kan men bijvoorbeeld het e-mailadres opvragen. Maar niet iedereen mag zomaar alle informatie van elk object bekijken of aanpassen, daar komt het auth-path bij kijken. Het auth-path moet ervoor zorgen dat clients de juiste rechten worden toegewezen zoals die in het systeem zijn beschreven. Het app-path dient voor speciale functies van bijvoorbeeld de webconsole die niet zomaar in de api gevat kunnen worden omdat ze extra acties vereisen. Voorbeelden hiervan zijn het herladen van een rekening, het maken van een betaling of het veranderen van zijn wachtwoord.

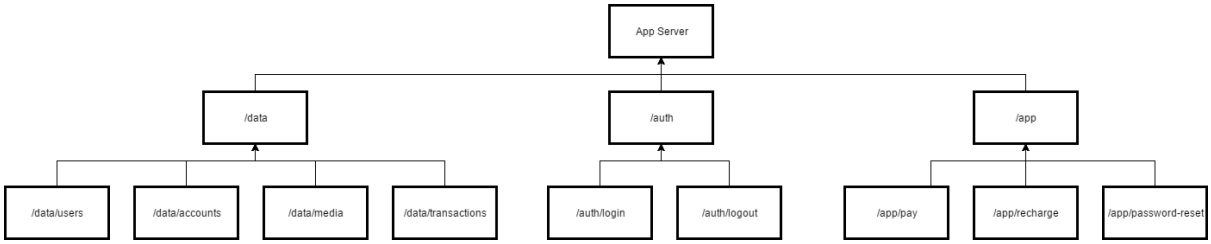


Fig. 11: Structuurschema appserver.

Data

Het data path van het betaalsysteem is een doorgeefluik van en naar de database server. Aangezien de database in vier tabellen met voor applicaties nuttige objecten is verdeeld, moeten er ook vier sub-paths onder /data komen. Eén voor gebruikers: ‘/data/users’, één voor rekeningen: ‘/data/accounts’, één voor fysieke identificatiemedia: ‘/data/media’ en een laatste voor transacties: ‘/data/transactions’. Voordat de appserver reageert met de juiste informatie van het juiste object moet er gecontroleerd worden of de client in kwestie wel de juiste rechten bezit. Dit doet de server door eerst na te kijken hoe de client de aanvraag wil authenticeren. Wanneer de client een gebruiker is gebeurd dat via een sessie, wat ik in auth bespreek. Wanneer een externe applicatie een aanvraag doet authenticeert die zich met een speciale api-key. Wanneer de betaalterminals van het systeem zelf de api gebruiken moeten die een speciale sleutel gebruiken. Met die informatie van een van die drie authenticatiemethoden moet de appserver nagaan welke rechten men heeft. Als de rechten voldoen, dan gaat de aanvraag door. De appserver moet dan kunnen reageren op verschillende acties die een client kan aanvragen. Men kan een object namelijk lezen, aanmaken, aanpassen of verwijderen. Aan de hand van die actie moet de appserver de databaseserver contacteren en ook respectievelijk dat object uitlezen, aanmaken, aanpassen of verwijderen.

Auth

Dit tweede path dient om gebruikers en beheerders te authenticeren met de server. Dit moet hoofdzakelijk gebeuren omdat het wachtwoord voor veiligheidsredenen niet bij de client mag worden opgeslagen. Om zich te authenticeren moet de client eerst de juiste gegevens zoals een gebruikersnaam en wachtwoord doorsturen (/auth/login). De server kijkt die gegevens dan na door de databaseserver te contacteren. Als ze juist zijn antwoordt de appserver met een sleutel die de client in de toekomst kan gebruiken. Ook terug uitloggen kan ondersteunt worden (/auth/logout).

App

Het derde en laatste path dient voor speciale functies die extra acties door de appserver vereisen. Een voorbeeld hiervan is maken van een betaling. Om een betaling te maken moet er niet alleen een object in de transactions-tabel aangemaakt worden maar moet de juiste rekening ook effectief aangerekend worden. Hetzelfde geldt voor het herladen van een rekening. Hierbij moet de payment-API gecontacteerd worden. Die API moet de bankrekening van de gebruiker crediteren. Bij de verandering van een wachtwoord moet het huidige wachtwoord van de gebruiker ook worden gecontroleerd. Elk van die functies krijgt een eigen path zoals ‘/app/pay’, ‘/app/recharge’ of ‘/app/password-reset’.

Deel C:
Implementatie

In dit hoofdstuk beschrijf ik in detail hoe het ontwerp van het digitaal betaalsysteem uit het vorige hoofdstuk in de praktijk kan geïmplementeerd worden. Daarnaast maak ik ook eigenlijke prototypes en schrijf ik software voor bepaalde aspecten van het systeem.

Zoals elk groot project heeft mijn betaalsysteem een codenaam nodig. Ik ben uiteindelijk voor de naam *DSpay* voor *Digital School Pay* gegaan. Bij die naam ontwierp ik ook een simpel logo om bijvoorbeeld als favicon van de website te gebruiken.

Voor projecten waarbij programmeren is betrokken is een *version control system* of VCS essentieel. Een VCS maakt werken met meerdere mensen of met meerdere versies van een software makkelijker. Hiervoor is de meest populaire en wereldwijd gebruikte - en dus evidente - keuze Git en GitHub. GitHub is een website waar men gratis online Git repositories kan opslaan. Alle ontwikkelde software voor DSpay heb ik ook ter beschikking gesteld op GitHub. De belangrijkste referenties over de repositories en demowebsites van *DSpay* zijn:

General:
GitHub repository: https://github.com/Ubipo/DSpay

Webserver:
Domain: http://dspay.ubipo.net
GitHub repository: https://github.com/Ubipo/DSpay-Web

Appserver:
Domain: https://dspay.herokuapp.com
GitHub repository: https://github.com/Ubipo/DSpay-Node

Aangezien heel DSpay een uitgebreid en complex systeem is zal ik in de toekomst hoogstwaarschijnlijk al veel aanpassingen aan de software gemaakt hebben, het hieronder volgende is dus een momentopname van het prototype-proces.

1. Identificatiemedia

Het identificatiemedium is het fysieke voorwerp dat een gebruiker gebruikt om een betaling te maken. Zoals beslist in deel B is dat voor ons een studentenkaart met NFC-tag. Aangezien mijn systeem geen nood heeft aan geavanceerde NFC-functies hangt welk soort NFC kaart het systeem gebruikt vooral af van het aanbod van de leverancier. *Offini cards*, de leverancier van studentenkaarten voor Scheppers, bied Mifare kaarten in een variëteit aan capaciteiten en toepassingen. De simpelste van hun kaarten is de 'MIFARE Classic 1K'. Zoals de naam suggereert heeft de tag een geheugen van 1024 bytes. Een NFC kaart als deze kan bedrukt worden, zoals nu al wordt gedaan met normale blanco kaarten. Ik vond een gelijkaardige MIFARE Classic 1K van *Antratek* voor mijn prototype. Ook alle andere elektronica nodig voor mijn prototype kocht ik ook op Antratek (https://antratek.be).

2. Front-end hardware

De front-end hardware is die apparatuur waarmee een gebruiker, de administratie en operatoren zullen interageren. Hieronder volgt hetzelfde schema van de front-end hardware voor terminal type 1 zoals al ontworpen in deel B, met nu de exacte componenten om dit in de praktijk te kunnen bouwen. Een samenvatting

van welke componenten verschillen tussen de verschillende types van terminal (TT1, TT2 en TT3) volgt naar het einde van dit hoofdstuk toe.

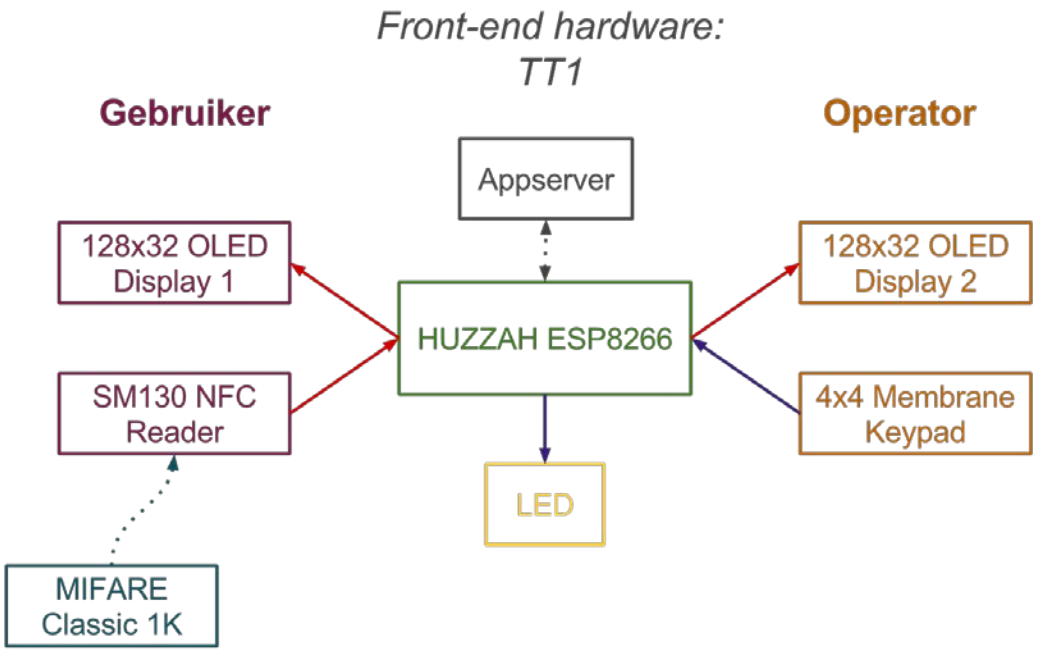


Fig. 12: Schema implementatie front-end hardware voor TT1.

2.1. De microcontroller

Het hart van elke betaalterminal is een microcontroller. Zoals al eerder beslist gebruiken we geen SOC-microcontroller, echter een simpelere minder op rekenkracht gerichte oplossing. Welke we precies kiezen hangt af van hoe de betaalterminal geïmplementeerd zal worden. In deel B ontwierpen we het systeem met twee soorten microcontrollers, een mét internetverbinding en een zonder.

Ik bespreek eerst de microcontroller voor TT2, zonder internetverbinding en dus verbonden met een computer. Voor zo'n microcontroller zijn er veel opties. Aangezien ik al door meerdere persoonlijke projecten ervaring heb met Arduino zal ik me alleen tot hun gamma beperken (www.arduino.cc). Moest dit systeem ook volledig uitgebouwd worden kan er uiteraard ook voor een ander type microcontroller gekozen worden aangezien de meeste microcontrollers in essentie niet veel verschillen.

Om te kiezen tussen de ongeveer tien bestande versies van Arduino microcontrollers gebruik ik vier vereisten: een lage prijs, compact ontwerp, genoeg rekenkracht en geheugen en zoals ontworpen, genoeg IO voor mijn toepassing. Met genoeg IO bedoel ik al het benodigde om een keypad, twee displays, NFC lezer en twee LED's aan te sturen. Bijna alle moderne displays hebben een I2C interface en kunnen dus gedaisychained worden. Dit samen met de seriële poort die voor een NFC-shield wordt gebruikt maakt vier digitale IO poorten. Tel daarbij nog twee poorten voor LED's en onze vereiste is zes digitale IO en één analoge input om de keypad via een resistor matrix uit te lezen. Genoeg geheugen betekent vooral geheugen om twee displays aan te sturen aangezien er een buffer van het hele display moet worden bijgehouden. De goedkoopste en meest compacte maar nog steeds voldoende rekenkrachtige optie met genoeg geheugen is dan de Arduino Pro Mini 328 aan maar €10 per stuk. Er bestaat een 3.3V/8MHz en een 5V/16MHz versie maar met de opkomst van 3.3v sensoren is die eerste versie naast efficiënter ook meer future-proof.

De tweede, geavanceerdere betaalterminal is die met een microcontroller die wel een internetverbinding heeft. Een systeem met internetverbinding kan op twee manieren worden opgebouwd. Oftewel met een externe WiFi-chip oftewel met een microcontroller met een ingebouwde WiFi-chip. Die eerste optie komt vaak in de vorm

van de ESP8266, wat in de wereld van de microcontrollers bijna een even groot hype-woord is geworden als IOT zelf. Maar aangezien veel nieuwe variaties op de 8266 zoals de esp-12, meer en meer IO's hebben is het niet meer nodig om een aparte microcontroller te hebben. Een totaalsysteem met WiFi en IO inbegrepen zoals de HUZZAH ESP8266 (een 3.3v breakout voor de esp-12 door Adafruit) is hiervoor uitermate geschikt. Met negen digitale io's en exact één analoge input kon ik geen beter board wensen.

2.2. Input

TT2 zal het toetsenbord op de computer waarmee die verbonden is gebruiken ne TT3 krijgt informatie over het te betalen bedrag via de automaat. Geen van beiden hebben dus een extra HID nodig. Terminal type 1 daarentegen heeft een eigen HID nodig om bedragen in te tikken.

Het meest budget vriendelijke en simpelste HID voor microcontrollers is een keypad. Keypads zijn naast snel ook simpel met een microcontroller te implementeren en goedkoop aan maar €7 voor een 4 x 4 membranen keypad. Zo'n membranen keypad heeft naast tien cijfers en twee extra toetsen (* en #) ook vier lettertoetsen. Die lettertoetsen kunnen gebruikt worden om bijvoorbeeld snel het bedrag voor een volledige maaltijd of dat voor een dessert in te geven. Dat het bovenvlak van zo'n keypad volledig afgesloten en dus waterdicht is, is een extra voordeel.

Keypads zijn dan wel simpel maar toch moeilijk aan te sluiten. Ze vergen namelijk acht IO's op de microcontroller, één voor elke rij en kolom. Gelukkig bestaat er iets als een *resistor matrix*.

Om de werking van een resistor matrix te begrijpen moet men echter eerst die van een keypad begrijpen. Zoals met onderstaande afbeelding wordt geïllustreerd zijn de vier rijen en kolommen van een zestien toetsen groot keypad verbonden door de schakelaars onder die toetsen. Als een toets wordt ingedrukt vormt er zich een verbinding tussen die specifieke rij en kolom. Voor toets '7' is dat bijvoorbeeld rij 3 en kolom 1. Voor toets '#' is dat rij 4 en kolom 3. In de praktijk en zoals op de linker figuur worden daarbij respectievelijk draden 3 en 5 voor '7' en draden 4 en 7 voor '#' verbonden.

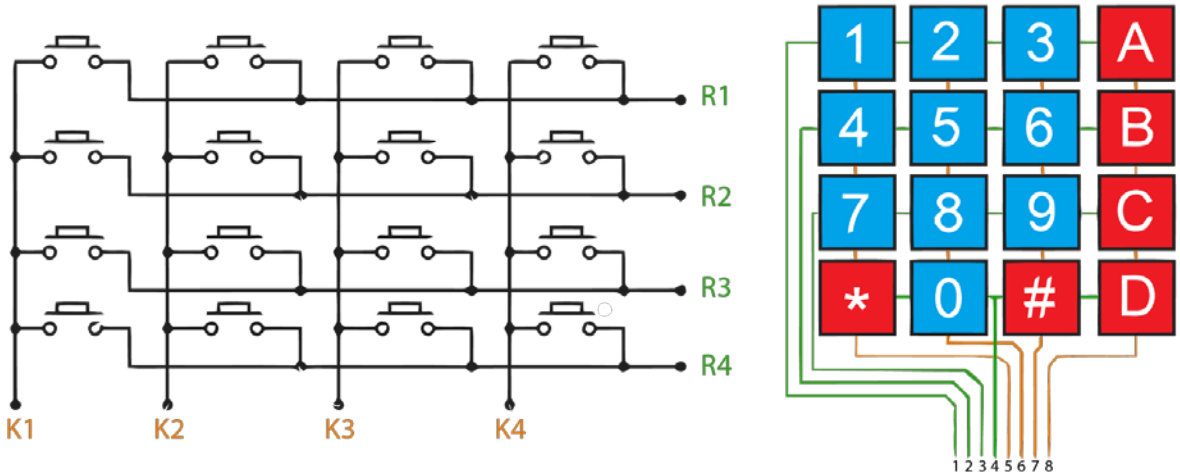


Fig. 13: Schema bedrading keypad.

Om na te kijken welke toets werd ingedrukt zou je het keypad moeten 'pollen'. Dit betekent telkens over elke rij een signaal sturen en nagaan of een van de kolom-draden een signaal teruggeeft. Aan de hand van welke kolom dat was en door welke rij het signaal op dat moment ging weet je welke toets werd ingedrukt. Dit zou voor een 4x4 matrix zoals gezegd acht digitale IO poorten in beslag nemen. Om dat op te lossen gebruikt men dus een resistor matrix. Hierbij wordt elke rij met weerstanden in serie verbonden. Zo wordt R1 via een weerstand met R2 verbonden en R2 via een weerstand aan R3 en zo verder. Voor de kolommen gebeurt

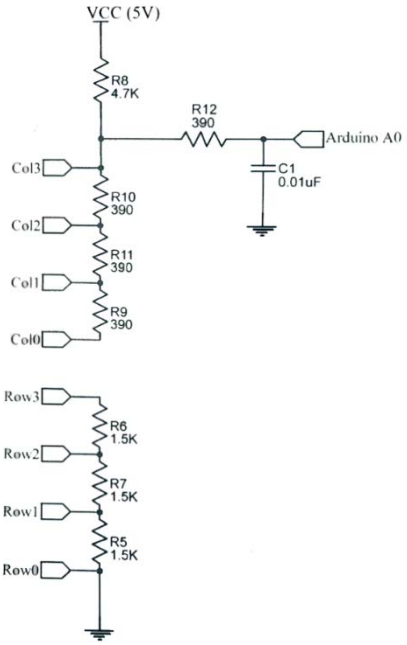


Fig. 14: Circuitschema resistormatrix.

hetzelfde. Als we dan R1 als de input van het resistormatrix en K1 als de output ervan beschouwen dan is de matrix één grote weerstand wiens waarde afhangt van welke toets er wordt ingedrukt (de waarde van weerstanden in serie kan worden opgeteld). Gecombineerd met een voltage divider kunnen we een variabel voltage krijgen dat afhangt van welke toets er op het keypad wordt ingedrukt. Een diagram van dat circuit is hier links te zien. Merk op dat 'R5' voor 'Resistor 5' naar een weerstand en niet naar een rij verwijst zoals dat bij de vorige twee illustraties het geval was. Op het diagram is ook 'Arduino A0' te lezen, dit is een analoge input van de Arduino. Via die poort kan het potentiaal (voltage) tegenover de grond uitgelezen worden en zo ook indirect welke toets werd ingedrukt.

2.3. Output

De output van de drie type terminals verschillen wat betreft de display voor de operator. Terminal type 1 heeft namelijk wel een display voor de operator maar TT2 of 3 niet. De display voor de gebruiker en LED blijft bij elk type terminal hetzelfde.

Displays voor microcontrollers zijn meestal oftewel LCD voor *liquid-crystal display* of *OLED* voor *organic light-emitting diode*. LCD's, zoals die onderaan, hebben een backlight nodig terwijl OLED's hun eigen licht genereren, waardoor die vaak beter leesbaar zijn. OLED's hebben ook het voordeel meer mogelijkheden te bieden als het gaat om pictogrammen en speciale karakters omdat ze moderner zijn. Daarnaast hebben ze ook vaker dan LCD's een I2C of SPI interface waardoor ze een stuk minder IO's innemen op de microcontroller. Antratek verkoopt voor €17,50 een 128x32 pixel I2C OLED display, rechts te zien. Dit is zowel voor mijn prototype als voor een eigenlijke implementatie een goede keuze aangezien de display genoeg ruimte biedt en met helder witte LED's makkelijk leesbaar is.

Fig. 16: Illustratie OLED display.

Fig. 15: Illustratie LCD.

3. Server stack

In deel B besprak ik de server stack van DSpay onder 'Back-end hardware' om het onderwerp dan nog simpel te houden. Maar een server stack is eigenlijk een mix van hard- en software en is een stuk geavanceerder dan hoe ik het toen uitdrukte. Er kunnen bijvoorbeeld meerdere servers op één machine draaien. En de servers die ik voor het prototype gebruik worden uiteraard niet door mij maar door cloud hosting providers gehost. Eén van de providers genaamd Heroku host bijvoorbeeld zowel de App- als de Databaseserver, wat onderscheid nog moeilijker maakt. Om al de relaties tussen cloud service, servers en machines duidelijk te maken, breidde ik het schema uit deel B hieronder nog een stuk uit.

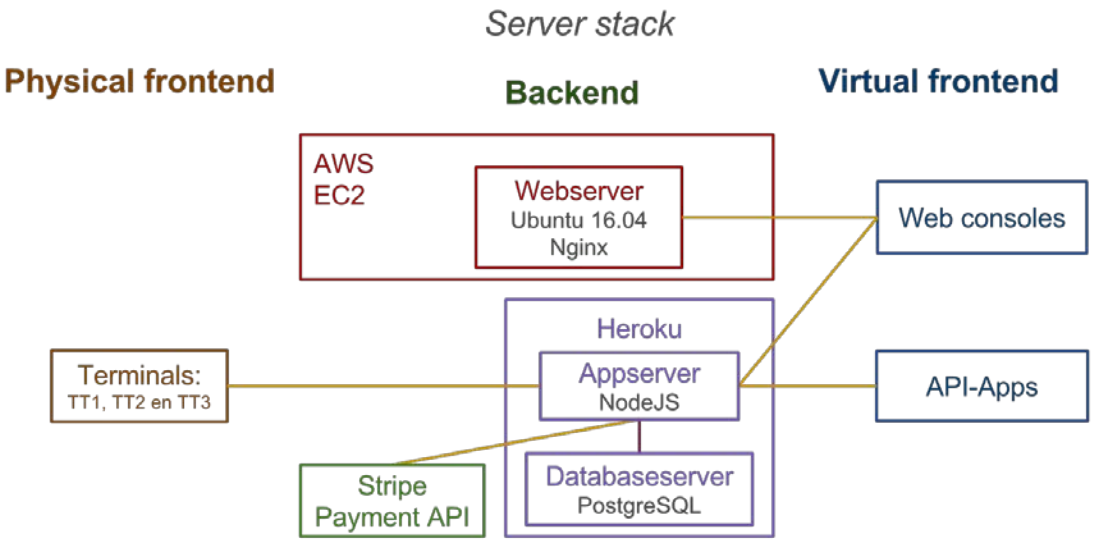


Fig. 17: Diagram geavanceerde server stack.

Net zoals in deel bij maak ik een onderscheid tussen de physical en de virtual front-end. Maar in dit diagram is het backend verdeeld in niet alleen servers maar ook services. De eerste daarvan zijnde AWS EC2.

3.1. AWS EC2

AWS, voor Amazon Web Services is een cloudplatform met allerhande services. Daaronder valt onder andere EC2, voor elastic compute cloud. Elastic computing betekent dat een server meer rekenkracht kan geboden worden naarmate dat nodig is. Zo bestaan er verschillende *tiers* van servers met verscheidene niveaus van rekenkracht en voor verscheidene toepassingen. In mijn geval is dat niveau 't2.micro', dus de tweede laagste tier, met maar weinig rekenkracht. Veel rekenkracht is er ook niet nodig voor servers voor computerspelletjes en een webserver voor persoonlijke projecten, de twee dingen waarvoor ik de server al een tijdje gebruik. Ook de webserver van DSpay vereist niet veel rekenkracht, waardoor AWS EC2 ideaal is om het prototype van de webserver op te laten draaien.

3.1.1 Webserver

Zoals op het bovenstaande schema te zien is, is de webserver met een gele lijn met de webconsoles verbonden. Gele lijnen in het schema duiden een HTTP verbinding aan. Ik gebruik voor de webserver dan ook een HTTP server, met name Nginx. Vergeleken met waarschijnlijk de populairste HTTP server, Apache, is Nginx lichter en meer gericht op *load balancing*. Dat tweede is voor mij niet erg van belang maar aangezien ik Nginx al eerder gebruikte zet ik hem nu ook in voor DSpay. Het overzetten van een Nginx configuratie naar eentje voor Apache is trouwens niet veel werk. De configuratie op zich is namelijk niet heel geavanceerd, maar dat bespreek ik onder hoofdstuk 4.1: Webserver.

3.2. Heroku

Heroku is net als AWS een cloudplatform. Het grote verschil tussen de twee is dat Heroku een *PaaS* of *Platform as a Service* is. Bij een PaaS moet men zelf niet bijvoorbeeld Nginx configureren of het onderliggende OS updaten. Dit is naast simpeler ook eenvoudiger te *scalen* wat zoveel betekent als elastic computing maar dan live, dus terwijl de server draait. Scalability is vooral handig voor de app- en databaseserver, die veel meer aanvragen zouden ontvangen dan de webserver.

3.2.1 Appserver

De belangrijkste server die op Heroku's platform draait is de appserver. De serversoftware die ik daarvoor gebruik is *NodeJS* (in sommige gevallen gewoon 'Node' genoemd). NodeJS is zoals de naam suggereert een variant van JS voor Javascript, maar dan voor de server-side. Voordelen van NodeJS over bijvoorbeeld Python of PHP zijn onder andere dat het dezelfde programmeertaal gebruikt als de client-side (front-end software: webconsole) en *asynchroon* werkt. Asynchrone executie betekend dat een programma niet wacht tot een proces - bijvoorbeeld een databaseaanvraag - is afgehandeld maar direct verdergaat met andere code. De appserver communiceert net zoals de webserver over hoofdzakelijk HTTP met de uitzondering van de databaseserver. Dat verkeer gaat over een eigen protocol. Welke extra libraries de appserver zal gebruiken en hoe die reageert op aanvragen bespreek ik onder back-end software: appserver.

3.2.2 Databaseserver

Ook de databaseserver draait op Heroku's platform. Dit is vooral zo omdat Heroku die service aanbiedt en die dus goed geïntegreerd is met de NodeJS server en met goed gedocumenteerd is. De databaseserver die ik zal gebruiken is PostgreSQL, een relationele database (met rijen en tabellen) en een object-relational database, wat betekend dat objecten in een bepaalde tabel kunnen verwijzen naar objecten in een andere tabel, zoals ik in deel B al aanhaalde. De connectie met de database verloopt over een eigen protocol en dus niet HTTP. Hoe de tabellen exact werken bespreek ik tijdens back-end software: databaseserver.

3.3. Stripe

Stripe is een payment API. De appserver gebruikt Stripe om gebruikers de mogelijkheid te bieden om hun rekening te herladen met een bankkaart of kredietkaart. Ik koos Stripe over bijvoorbeeld Braintree of PayPal omdat ze excellente documentatie en relatief lage transactieprijzen bieden. Een transactie met een Europese bankkaart kost €0.25 en 1.4% op het betaalde bedrag. Strikt gezien maakt Stripe geen deel uit van de server stack omdat het slechts een API is en geen server die ik programmeer, daarom bespreek ik het dus niet tijdens Back-end software.

4. Back-end software

4.1. Webserver

In dit hoofdstuk bespreek ik de enige configuratie die nodig is voor de webserver namelijk die voor Nginx.

Elk configuratiebestand van Nginx heeft een *serverblock*. Een serverblock beschrijft hoe Nginx moet reageren op een bepaald subdomain, in ons geval dus 'dspay.ubipo.net'. Een serverblock zelf is vaak verdeeld in *locations*, wat praktisch overeenkomt met het path in een URL. Voor het prototype zijn er maar twee simpele locations nodig, de eerste zijnde '/'.

De root location, aangeduid met '/' in Linux en op het web is de absolute basis. Wat er in deze location staat beschreven voert Nginx dus uit wanneer er geen path wordt gegeven en een gebruiker bijvoorbeeld gewoon 'http://dspay.ubipo.net' intikt. Deze locatie moet dus de thuispagina genaamd 'index.html' teruggeven. Maar aangezien de website dynamisch is en dus niet echt met verschillende pagina's werkt en telkens hetzelfde kader moet Nginx altijd index.html teruggeven. Wanneer een client bijvoorbeeld naar

‘http://dspay.ubipo.net/transactions’ gaat moet de server ook index.html teruggeven. De javascript binnen index.html vult de pagina dan aan met de juiste informatie over transacties. Dit gebeurt zo:

```
location / {
	root /home/ubuntu/DSpay/web;
	try_files /index.html =404;
}
```

‘root’ geeft aan waar Nginx moet zoeken naar bestanden en ‘try_files’ zegt dat alleen index.html teruggegeven moet worden. Als index.html teruggegeven niet lukt omdat dat bestand er bijvoorbeeld niet staat moet de server HTTP error 404 (de code ‘404’ betekent ‘not found’) teruggeven.

4.2. Appserver

Zoals tijdens deel B besproken moet de appserver reageren op de paths ‘/data’, ‘/app’, ‘/auth’ en hun subpaths. Om die functionaliteit te bieden heeft de NodeJS applicatie enkele *packages* nodig. Packages zijn extensies of modules voor NodeJS die de functionaliteit vergroten. De belangrijkste package is *Express*. Express is een framework. Een NodeJS framework biedt basisfunctionaliteit voor webapps zoals gebruikerssessies en *routing*. Routing is het verdelen van de server in paths. Een tweede belangrijke package is *Knex.js*. Knex maakt werken met een postgres database makkelijker.

Voor authenticatie van gebruikers gebruik ik de Passport.js package. Passport zorgt ervoor dat gebruikers zich kunnen authenticeren met de server. Met Passport is het makkelijk om bijvoorbeeld later ook andere manieren van authenticatie zoals met facebook toe te laten. Maar met Passport alleen kunnen gebruikers zich nog niet met een gebruikersnaam en wachtwoord inloggen. Daarvoor dient passport-local, een extensie van Passport

4.2.1 Data

Wat de server moet terugsturen wanneer een client een verzoek doet op het data path is makkelijk te definiëren. Dit komt omdat ik gebruik maak van een zogenaamde *REST*-structuur. REST is een manier van ontwerp voor API’s. Een *RESTful* API moet gebruikmaken van ten eerste een standaard *HTTP verb*, ten tweede van standaard HTTP URL’s en ten derde van een standaard datastructuur.

HTTP verbs zijn een soort code in elk HTTP verzoek die de server vertellen welke actie de client wil ondernemen. Het HTTP verb ‘GET’ bijvoorbeeld, betekent dat de client een object wil opvragen. Het verb ‘POST’ vertelt de server dat de client een nieuw object wil aanmaken. ‘PUT’ dient om een object met nieuwe informatie te voorzien.

Hoe men een standaard URL samenstelt is simpel te begrijpen door aan een mappenstructuur te denken. Neem bijvoorbeeld een catalogus van alle dierensoorten. Wanneer de URL dan ‘/’ - ook wel ‘root’ - is, dan verwacht men een verzameling van alle dieren. Wanneer men ‘/vertebraten’ opvraagt verwacht men alle vertebraten. En in ‘/vertebraten/warmbloedige’ alle vogels en zoogdieren. Voor de appserver betekent dit dat wanneer men ‘/users’ opvraagt men een lijst van alle gebruikers krijgt (als men de juiste authenticatie heeft uiteraard). Wanneer men ‘/users/Pieter.Fiers’ opvraagt verwacht men alle informatie over de gebruiker Pieter Fiers.

Een standaard datastructuur is bijvoorbeeld JSON, voor Javascript Object Notation. Bij JSON is het bijvoorbeeld de afspraak dat objecten binnen gekrulde haakjes staan waarbij de naam van het object en de waarde met een dubbel punt zijn gescheiden zoals hier: {naam: Jan}. Zo’n afgesproken datastructuur maakt dat client en server elkaar makkelijker begrijpen.

Omdat alle verschillende REST-aanvragen voor elk soort object (users, transactions, media, accounts) hetzelfde zijn zal ik die maar toepassen op één soort object namelijk users.

GET /data/users

Dit is de simpelste REST-aanvraag. Wanneer de NodeJS server een aanvraag krijgt met deze structuur moet die eerst de authenticatie van de client nakijken. Daarvoor leest de code oftewel de session_token of de api_key die de client meestuurde uit. Stuurde de client een session_token dan moet de code die simpelweg in een functie van Express voeren waarna Express 'true' of 'false' teruggeeft. Geeft Express true aan dan kan het programma verder. Antwoord Express false dan stuurt NodeJS een HTTP errorcode 401: 'unauthorized. Bij een api_key kijkt NodeJS die key na in de 'api_keys' tabel via Knex. Waarna de ook oftewel door kan lopen oftewel 401 kan terugsturen. Als de gebruiker inderdaad gemachtigd is dan kan de server een Knex-functie zoals hieronder uitvoeren:

```
knex.select('*').from('users')
```

Deze Knex functie geeft zoals we willen alle gebruikers terug uit de user tabel. Dit kan nuttig zijn voor bijvoorbeeld de webconsole voor administratie wanneer men een tabel van alle gebruikers wil zien. Voor de gebruikersconsole zou een functie zoals deze nuttiger zijn:

```
knex('users').where({
      uid: 'ses_uid''
}).select('username')
```

Die functie vraagt voor een bepaalde user ID uit de usertabel de gebruikersnaam op. Heeft NodeJS dat gedaan dan wordt die informatie teruggestuurd naar de client.

Deze structuur kan men nu ook toepassen op bijvoorbeeld 'GET /data/transactions' om alle of specifieke transacties op te vragen.

PUT /data/users

Deze REST-aanvraag dient om een object, in dit geval een gebruiker de updaten. Het eerste wat de NodeJS server moet doen is hetzelfde als bij 'GET /data/users' alleen moet een gebruiker of api_key nu ook gemachtigd zijn om een wijziging aan te brengen aan het object in kwestie. Als dat zo is doet voert NodeJS weer een Knex functie uit alleen deze keer om een object te wijzigen. Voor de gebruikersconsole zou dat bijvoorbeeld de persoonlijke instellingen van een gebruiker kunnen zijn. Gebeurt dit succesvol dan geeft NodeJS een HTTP code 200: 'OK' door.

Voor mijn prototype is het niet nodig dat gebruikers zelf nieuwe gebruikers zouden kunnen aanmaken of verwijderen daarom bespreek ik ook alleen GET en PUT.

4.2.2 App

Het '/app' path dient zoals we uit deel B weten voor speciale acties zoals een betaling maken. Voor mijn prototype bespreek ik dus de essentiële '/app/pay', '/app/recharge' en '/app/pwd-reset'. Deze NodeJS paths moeten niet voldoen aan REST specificaties en worden dus allemaal via POST aanvragen gedaan.

/app/pay

Het path '/app/pay' dient logischerwijs om een betaling te maken. Wanneer een client dus in JSON formaat een aanvraag doet op dit path moet de NodeJS server enkel dingen doen. Ten eerste moet ze alle informatie uit de aanvraag halen. Bij bijvoorbeeld deze aanvraag:

```
{
      "value": "€5",
      "description": "Grote smos.",
      "account": "a5b99c5e",
}
```

…is de waarde uiteraard €5 en descriptie 'Grote smos' en de rekening ID is "a5b99c5e". De tweede stap is om die rekening na te kijken. Daarvoor gebruikt het programma dus Knex en vraagt zo de balans van de rekening

met ID “a5b99c5e” op. Is die balans positief dan wordt het bedrag van de rekening afgetrokken en wordt er een nieuwe transactie aangemaakt in de transactie tabel. NodeJS reageert dan op de aanvraag met HTTP code 200: ‘OK’. Gaat de balans onder de nul maar nog niet onder de vooraf vastgelegde drempelwaarde dan doet NodeJS hetzelfde als hiervoor maar stuurt nog extra data met het antwoord om de client te waarschuwen dat de rekening een lage balans heeft. Gaat de balans door de betaling onder de drempelwaarde dan wordt de betaling geweigerd met HTTP error code 401.

Maar een terminal moet ook een betaling kunnen laten doorgaan als de balans onder nul gaat, anders zou een leerling bijvoorbeeld geen middagmaal kunnen betalen. Hiervoor moet de client (bijvoorbeeld TT1) een extra object “force”:”true” met de aanvraag sturen. Als dit door een client wordt gedaan gaat de betaling sowieso door, ook al is de balans te laag. Uiteraard is niet elke client gemachtigd om zo’n betaling door te voeren.

/app/recharge

Gebruikers moeten via de gebruikersconsole ook hun rekening kunnen bijvullen. In het prototype voor de gebruikersconsole nog niet geïmplementeerd maar wel al bij de appserver. Het bijvullen van een rekening gebeurd via het path ‘/app/recharge’. Bij een aanvraag op dit path moet de client ofwel een token ofwel nieuwe betaalgegevens doorsturen. Als een gebruiker al eerder een rekening herladen heeft krijgt die in de user tabel een payment token. Deze token kan dan gebruikt worden om in de toekomst makkelijker de rekening te herladen. Wanneer het om de eerste betaling van een gebruiker gaat moet de client alle betaalinformatie zoals een creditcardnummer en CVC doorsturen. NodeJS houd deze informatie om veiligheidsredenen niet zelf bij maar stuurt die door naar Stripe. Stripe stuurt dan een payment token terug en informatie over of de transactie al dan niet succesvol was. Die payment token wordt opgeslagen en er wordt via Knex een nieuwe transactie aangemaakt in de transactions tabel. Nadat ook de balans van de rekening is geüpdatet stuurt NodeJS een antwoord op de aanvraag van de client met HTTP code 200: ‘OK’ als de betaling succesvol was.

/app/pwd-reset

Dit path gebruik ik in het prototype van de gebruikersconsole. Het dient zoals de naam suggereert om een wachtwoord van een gebruiker te veranderen. Na authenticatie van de aanvraag veranderd NodeJS daarvoor simpelweg het wachtwoord van de gebruiker in de users tabel.

5. Front-end software

Tijdens de implementatie van de front-end software zal ik me vooral concentreren op de software voor de betaalterminals zelf en niet bijvoorbeeld die van de webconsole die u, de lezer, zelf in actie kunt zien op http://dspay.ubipo.net.

5.1. Terminals

In de volgende drie hoofdstukken bespreek ik in detail de werking van de software die op TT1, TT2 en TT3 zal draaien. Software voor de Arduino en Arduino compatibele microcontrollers zoals de HUZZAH esp-12 wordt geschreven in een variant van C++. Die code moet door de Arduino IDE interpreter gecompileerd worden en kan dan geüpload worden naar de microcontroller.

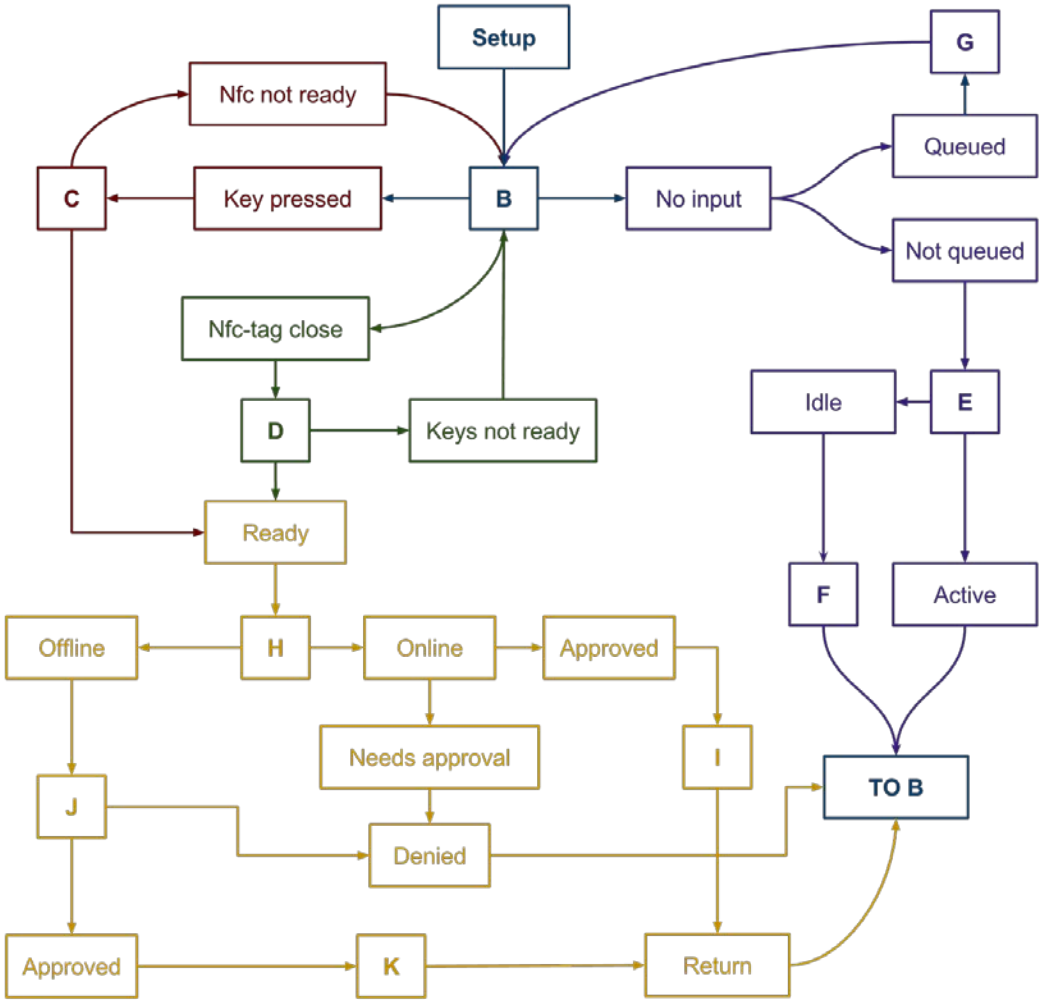


Fig. 18: Flowchart software TT1.

Setup:
Dit is een deel van de code dat altijd wordt uitgevoerd bij het opstarten van een Microcontroller. Tijdens de setup wordt er gedefinieerd welke poorten voor in- of output worden gebruikt, wordt er verbinding gemaakt met een WiFi netwerk (voor online terminals) en worden de display(s) getest. Hierna wordt functie *F: Heartbeat* uitgevoerd.

B: Main loop:
Wanneer die setup is doorlopen begint het programma de main loop uit te voeren. De main loop is de constante cyclus die de Microcontroller doorloopt als die met geen taak in het bijzonder bezig is. De Arduino kiest hier tussen drie routes. Ten eerste kan er een toets ingedrukt worden op de keypad. In dat geval springt de code naar functie *C: Keypress*. De tweede optie is dat er een NFC-tag in de buurt van de lezer is. In dat geval springt de code naar *D: NFC close*. Het laatste geval is dat er niets gebeurd. In dit geval springt de code naar functie *E: Idle check* of functie *G: Queue* aan de hand van de variabele 'queued'.

C: Keypress
De Keypress functie slaat de toets die werd ingedrukt op en update het display van de operator met het juiste karakter (bijvoorbeeld '5' of 'B'). Is die toets een speciaal teken 'A', 'B', 'C', 'D', '#' of '*', dan slaat de Arduino 'True' op in 'Keys_ready'. Dit betekend dat de operator klaar is met gegevens intypen. Is de toets een cijfer dan

slaat de Arduino 'False' op in 'Keys-ready'. Hierna wacht de Arduino tot de toets in kwestie niet meer wordt ingedrukt en springt dan afhankelijk van de variabele 'NFC_ready' terug naar *B: Main loop* of naar *H: Ready*.

D: NFC close
Deze functie dient om gegevens van de NFC-tag uit te lezen. Daarvoor spreekt de Arduino de NFC-reader aan en leest zo de eerste twee *Blocks* van de NFC-tag uit. Als dit lezen succesvol was wordt een LED kort aangezet om feedback te geven aan de gebruiker. Het programma haalt dan uit de eerste block haalt de UID van de NFC-tag en uit de tweede block het secret. Die twee gegevens worden opgeslagen in hun respectievelijke variabele en in een verzoek verstuurd naar '/data/media'. De appserver reageert hierop met gegevens over de NFC-tag en de geassocieerde rekening zoals de naam en balans. Die gegevens worden voor zowel operator als gebruiker weergegeven op hun display. Reageert de appserver niet dan worden er geen gegevens over de gebruiker weergegeven. Hierna springt de Arduino afhankelijk van de variabele 'Keys_ready' oftewel naar *B: Main loop* of naar *H: Ready*.

E: Idle check
In deze simpele functie kijkt de Arduino na of het programma al een lange tijd niet meer actief is geweest. Dit gebeurd door eerst de variabele 'Idle_start' af te trekken van de huidige 'millis' waarde en die uitkomst met een constante 'Idle_time' te vergelijken. 'Idle_start' houdt bij wanneer er voor het laatst een verzoek werd gestuurd naar de appserver. 'Millis' houd de tijd bij sinds dat de arduino is opgestart. Als het verschil tussen 'idle_start' en 'millis' groter is dan 5 minuten dan springt de functie naar *F: Heartbeat*, anders gaat ze terug naar *B: Main loop*.

F: Heartbeat:
Heartbeat is een functie die de connectie met de appserver nakijkt. Zo kan de operator op tijd worden gewaarschuwd dat de connectie met de server is onderbroken. De functie verzend dus een verzoek naar '/app/heartbeat'. Wanneer de appserver niet binnen de twee seconden reageert wordt het display van de operator met een icoontje geüpdatet dat aangeeft dat de terminal offline is. Als de server wel op tijd reageert springt het programma terug naar *B: Main loop*. Maar voor dat dat gebeurt wordt de variabele 'Idle_start' naar de huidige 'millis' waarde gezet: `< Idle_start= millis() >.` Zo weet het programma wanneer de laatste heartbeat verzonden werd.

G: Queue:
De queue functie zorgt ervoor dat aanvragen die offline opgeslagen werden verzonden worden naar de appserver. Hoe offline aanvragen worden opgeslagen bespreek ik in *L: Offline payment*. Aanvragen die door *K: Offline approval* zijn opgeslagen worden door de queue functie eerst uitgelezen en dan verzonden naar de appserver op '/app/payment'. Reageert de server op die aanvraag dan wordt de variabele 'queued' naar 'False' gezet om aan te geven dat er geen gegevens meer gequeued zijn. De queue wordt ook leeggemaakt en de functie springt terug naar **B.** Reageert de server niet dan worden er geen variabelen aangepast en springt de functie ook gewoon naar **B: Main loop**.
H: Ready
Deze functie wordt pas uitgevoerd wanneer de terminal alle informatie zoals het juiste bedrag en het NFC-secret en UID verzameld heeft. De Arduino verzendt die gegevens nu naar '/app/payment'. De appserver kan dan reageren met 'approved', 'warning' of 'denied'. Reageert de server met 'approved', dan springt de code naar I. Reageert de appserver met 'warning' dan springt ze ook naar *I: Online payment* maar met een speciaal argument. Reageert de server met 'denied' dan wordt *J: Online approval* uitgevoerd.

I: Online payment
In deze functie wordt feedback gegeven aan de gebruiker en operator door het oplichten van de LED en het updaten van de displays om aan te geven dat de betaling succesvol was. De functie kan ook uitgevoerd met een speciaal argument. In dat geval toont het display de gebruiker dat het saldo op zijn rekening laag is.

J: Online approval
Deze functie geeft aan operatoren de mogelijkheid om een betaling door te voeren ook al is het saldo van de gebruiker negatief. Hiervoor wordt er eerst een verzoek gedaan naar de appserver om extra informatie over de

rekening op te vragen. Het verzoek naar '/data/accounts' wordt door de server beantwoord met informatie zoals de laatste betaling en herlading op de rekening. Deze gegevens worden aan de operator getoond en geven de mogelijkheid om de betaling oftewel te laten doorgaan of te laten weigeren. Daarvoor wordt het keypad uitgelezen (polling) voor toetsaanslagen. Wanneer '*' of '#' worden ingedrukt wordt de betaling respectievelijk geweigerd of doorgevoerd. Wordt hij geweigerd dan geeft de LED en het display van de operator en gebruiker daar feedback over en springt de code terug naar *B: Main loop*. Wordt de betaling toch doorgevoerd dan verzendt de Arduino nog een verzoek '/data/accounts' alleen deze keer met een speciaal argument dat aangeeft dat de betaling sowieso moet doorgaan.

K: Offline approval
Deze functie geeft net zoals functie *J: Online approval* de operator de keuze om een betaling te weigeren of door te voeren. Verschillend van *J: Online approval* wordt hiervoor geen extra informatie van de server opgehaald. Ook in deze functie wordt het keypad voor '*' of '#' gepolled. Wordt de betaling geweigerd dan wordt dezelfde feedback als bij **J** gegeven en springt de functie terug naar *B: Main loop*. Wordt de betaling doorgevoerd dan wordt *L: Offline payment* uitgevoerd. Wanneer de connectie met de server in de toekomst dan wordt hersteld worden die gegevens in functie *G: Queue* alsnog naar de server doorgestuurd.

L: Offline payment
Deze functie slaat betalingen offline op om in de toekomst naar de appserver te verzenden. Dit gebeurd door eerst de variabele 'queued' naar 'True' te veranderen en dan alle betaalgegevens op te slaan in de queue. De operator en gebruiker krijgen ook feedback over het doorgaan van de betaling.

5.2. Webconsole gebruikers

In dit hoofdstuk bespreek ik eerst het ontwerp en dan de achtergrondsoftware voor de webconsole voor gebruikers. Ik breng dus de wireframes uit deel B tot leven in een echte website waarvan men het prototype kan bezoeken op dspay.ubipo.net. Voor zowel het ontwerp als voor achtergrondtaken moet ik een programmeertaal kiezen. Aangezien ik zelf nog maar weinig ervaring heb met PHP en omdat Javascript veel gebruikt is op het web koos ik voor het klassieke trio HTML, CSS en JS.

Om het web-development nog een stuk makkelijker te maken gebruik ik enkele libraries. De eerste hiervan is het webdesign-framework *Bootstrap*. Bootstrap maakt het ontwerpen van responsieve en moderne websites makkelijker met een platform bovenop het de standaard HTML en CSS-functies.
De tweede belangrijkste library is *jQuery*. jQuery is een essentiële javascript library die code simpeler en korter maakt.
Een derde library is *Handlebars*. Handlebars geeft de mogelijkheid om een soort sjabloon of *template* te maken voor een HTML pagina die je dan kunt invullen met de juiste data. Zo is het makkelijk om de website dynamisch te maken. Een dynamische website laadt sneller en gebruikt geen onnodige bandbreedte als een client hem opvraagt.
Andere libraries zijn *bootstrap-notify* voor handige en mooie notificaties, *animate.css* voor snelle animaties, *nprogress* voor een simpele progressbar en *bootstrap-dialog* voor slimme dialoogvensters.

5.2.1 HTML & CSS

Bij het ontwerp van de webconsole concentreerde ik eerst op het kader rond de inhoud. Dit kader bestaat uit de menubalk met navigatie en login-info bovenaan en de footer met extra informatie onderaan. Hierna heb ik respectievelijk de thuispagina, de transactiepagina en de accountpagina ontworpen.

Kader

Ik begon voor dit ontwerp met de navigatiebalk. Een navigatiebalk wordt in standaard HTML tags aangeduid met een `<nav>` tag met daaronder een lijst van links. Om zo'n normale navigatiebalk in een bootstrap *navbar* te veranderen moet men gewoon enkele HTML classes toevoegen. Voor de `<nav>` tag zijn dat `navbar`, `navbar-fixed-top` en `navbar-inverse`. `Navbar-fixed-top` geeft aan dat de navbar niet mee moet bewegen wanneer een gebruiker op een lange pagina naar beneden scrolt. `Navbar-inverse` geeft de navbar een aangenaam donker thema. Binnen de `<nav>` tag zitten drie elementen, namelijk de header en twee lijsten, één met de knoppen aan de linkerkant en één voor die aan de rechterkant. De header bevat de 'DSpay' tekst links bovenaan. De linkse lijst bevat de link 'Home': `<a id='nav_home'>Home</a>`, 'Transactions' en 'Account'. De lijst heeft ook de HTML id `nav_list` om later in de javascript-code te gebruiken. De rechtse navbar-lijst heeft twee elementen: de naam van de gebruiker die momenteel is ingelogd en een duidelijk uitlogknop. De uitlogknop gebruikt een handige functie van Bootstrap genaamd *glyphicons*; een soort pictogrammen voor veelgebruikte acties.

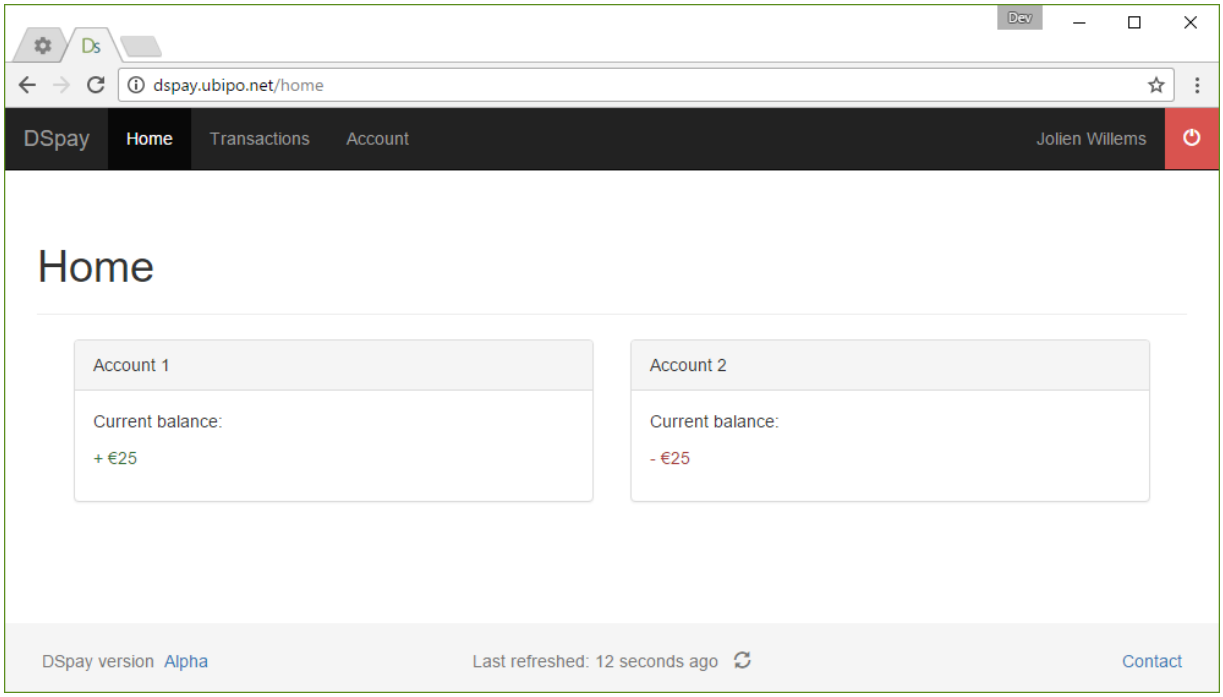


Fig. 19: Webconsole gebruiker 1.

De footer onderaan is spijtig genoeg niet in Bootstrap geïntegreerd. In tegendeel tot het fixed `<nav>` -element maakte ik de `<footer>` *absolute*. Dit betekend dat de footer wel zal meebewegen met de pagina. De footer is opgedeeld in drie gelijke delen. Eén voor de versie van de webconsole, één voor de algemene status en één voor bijvoorbeeld feedback of contactgegevens. Tijdens de prototypefase (Alpha) heb ik van het versienummer een handige link gemaakt naar het GitHub repository. De status in het midden geeft aan hoe lang geleden de nieuwste informatie werd opgehaald van de appserver en geeft de optie om manueel nieuwe informatie op te halen.

Thuispagina

De thuispagina moet zoals tijdens het ontwerp uit deel C de verschillende rekeningen van een gebruiker weergeven. In deze fase heb ik nog geen grafiek of voorspelling toegevoegd maar enkel de balans op de rekeningen. De pagina is opgedeeld in Bootstrap's panelen, aangeduid met de HTML-class `panel`. Door de pagina zo op te delen en Bootstrap's zogenoemde 'Grid system' te gebruiken is de pagina responsief voor kleine schermen, hiernaast geïllustreerd. Deze weergave illustreert ook de responsieve navbar. Wanneer het scherm te klein wordt voor een volledige navbar schakelt Bootstrap over op een uitklapbare versie.

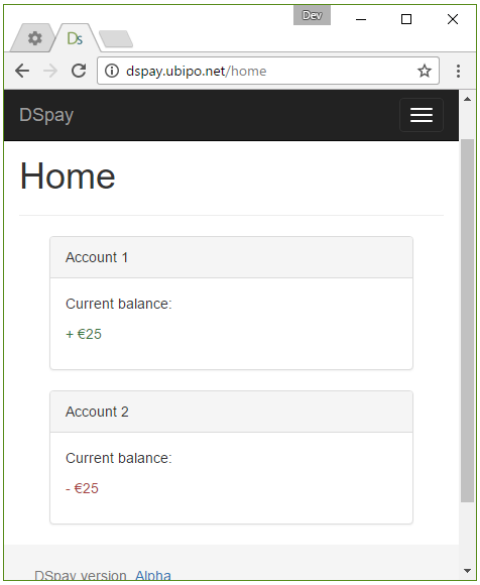


Fig. 20: Webconsole gebruiker 2.

Transactiepagina

De transactiepagina moet alle details weergeven van de laatste transacties. Hiervoor gebruik ik de HTML-tag `table`, met zoals bij de navbar een paar extra Bootstrap-classes om de tabel mobielvriendelijk te maken. Maar zelfs op een groot scherm kan niet alle informatie in verband met een transactie weergegeven worden. Daarom voegde ik een uitklapbaar paneel toe met extra informatie, zoals hieronder getoond. Het *Payed by* attribuut bij een transactie kan een fysiek identificatiemedia zoals een studentenkaart of een gebruiker zijn bij bijvoorbeeld het herladen van de rekening.

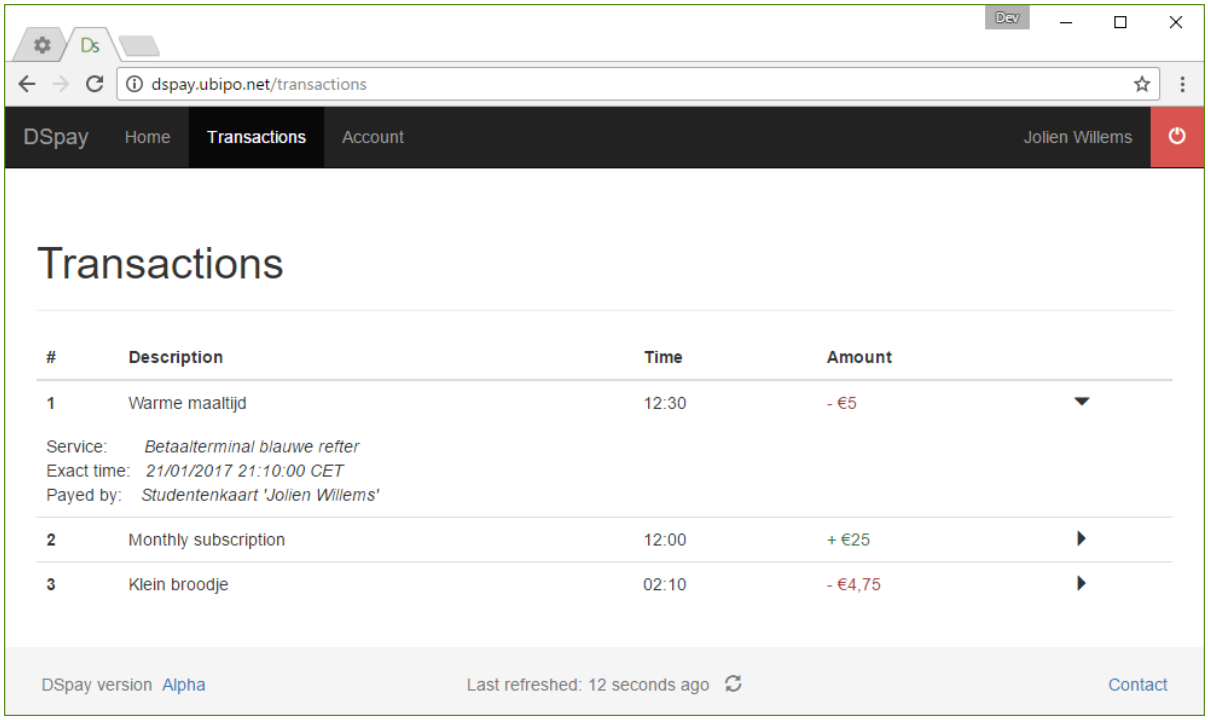


Fig. 21: Webconsole gebruiker 3.

Accountpagina

De accountpagina heeft nu nog niet alle functionaliteit die ik in deel B had ontworpen maar geeft wel al de mogelijkheid om een ander wachtwoord in te stellen en om API-Keys te beheren. API-Keys maken het mogelijk om andere programma's met DSpay te laten connecteren. In het eerste screenshot hieronder was ik de API-functionaliteit in het kader van *Smakelijk* aan het testen. Smakelijk, een dienst om via een app broodjes te bestellen (eindwerk van een andere Shepperiaan), gebruikt de API bijvoorbeeld om betalingen te maken. Het

tweede screenshot illustreert de functionaliteit van de bootstrap-dialog library met een dialoog voor het opnieuw instellen van zijn wachtwoord.

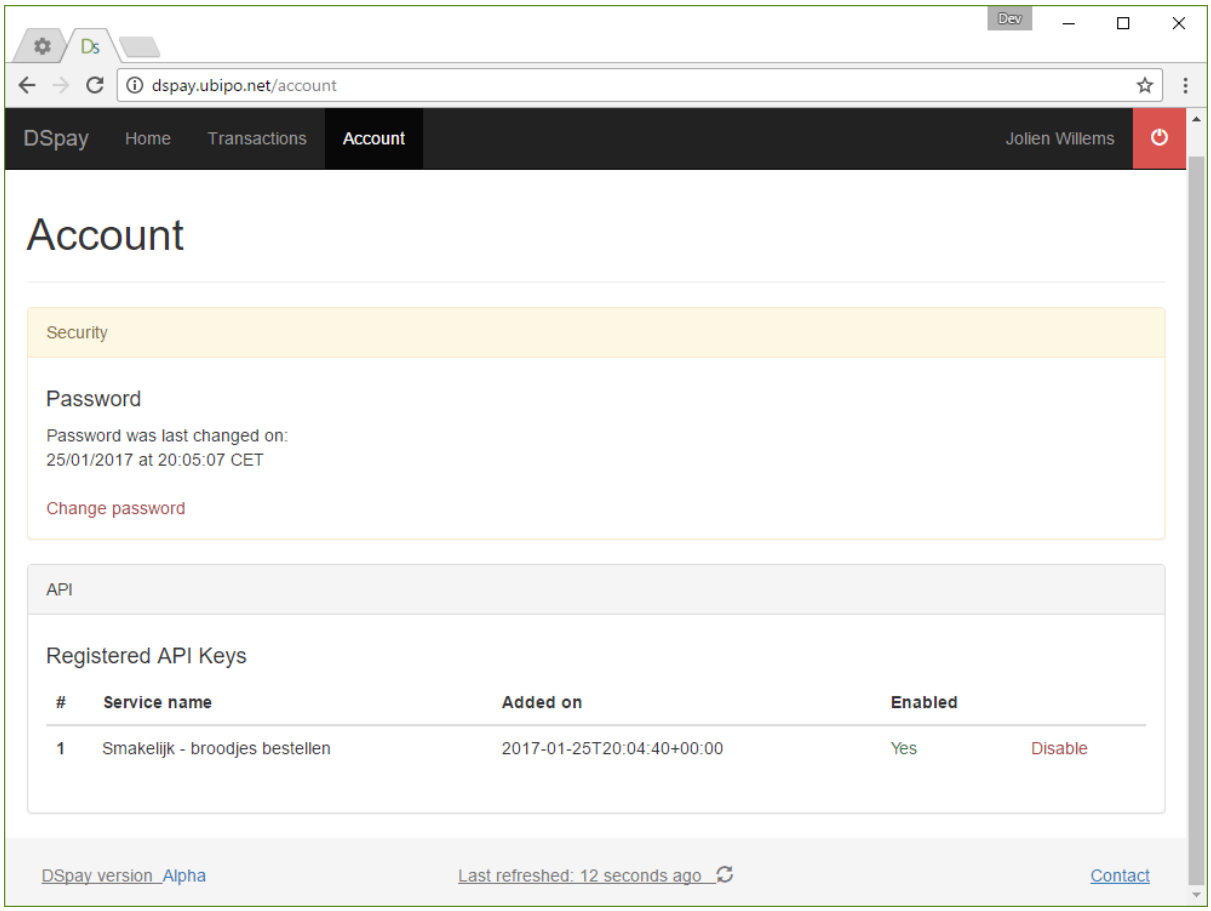


Fig. 22: Webconsole gebruiker 4.

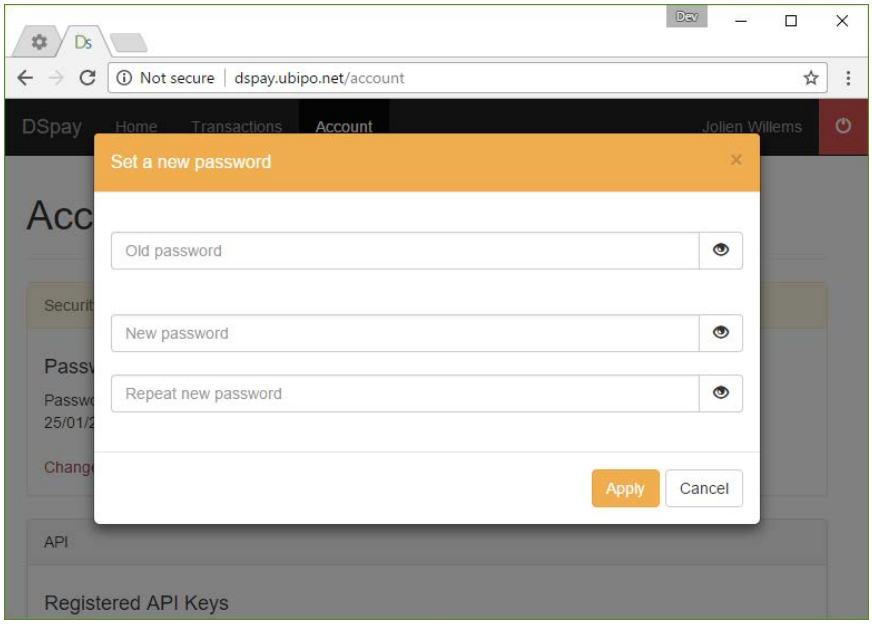


Fig. 23: Webconsole gebruiker 5.

5.2.2 Javascript

Maar al de functionaliteit zoals het weergeven van pagina's en wachtwoordherstel kunnen niet met enkel HTML en CSS gebeuren. Er is ook een zogenoemde scripting language nodig om op de achtergrond taken uit te voeren, voor mij Javascript en jQuery. Deze code is gelukkig niet zo geavanceerd als bijvoorbeeld die voor de betaalterminals maar is toch verdeeld in enkele functies. De eerste zijnde het laden van pagina's.

Navigate_to

De navigate_to functie brengt verschillende taken onder een functie die als argument een pagina om naar te navigeren heeft. De eerste stap is om een laadbalk te starten om de gebruiker duidelijk te maken dat er iets aan de gang is op de achtergrond. Met de *nprogress* library is dit makkelijk gedaan door `NProgress.start();`. Hierna doet de javascript een zogenaamde *ajax call* naar de webserver voor het juiste Handlebars-bestand (.hbt) voor de pagina waarnaar de gebruiker wil navigeren. Ajax, voor Asynchronous JavaScript and XML dient om op de achtergrond asynkroon bestanden op te halen. In ons geval is dat bestand geen XML maar een HTML (met .hbt als extensie), wat op zich niet uitmaakt. De javascript code voert ook de functie *Data_update* uit om de pagina met de nieuwste data van de api server te voorzien. Wanneer zowel de ajax call voor het pagina-template als de functie *Data_update* klaar zijn wordt de pagina door handlebars opgevuld met de juiste data, en komt er wordt er een nieuwe pagina toegevoegd aan de browsergeschiedenis. Dit laatste zorgt ervoor dat gebruikers heen en weer kunnen navigeren tussen pagina's, ook al worden er niet echt nieuwe pagina's geladen.

Data_update

Deze functie, gelijkaardig aan *navigate_to* maakt ook een ajax call, alleen deze keer naar de data-, in plaats van de webserver. Deze ajax call wordt door de server beantwoord met de data die voor die pagina nodig is. Die data wordt dan door een Handlebars-functie in de webpagina geplaatst. Deze functie kan ook met de herlaadknop aan de onderkant van elke pagina uitgevoerd worden.

Tot slot

Hopelijk heb ik u, de lezer, iets kunnen bijleren over webontwikkeling en ook over de vroegere betaalsystemen op het Scheppersinstituut. Ikzelf heb niet alleen kennis gemaakt met nieuwe technieken en systemen, ik heb ook ervaring opgedaan die misschien van pas zal komen bij mijn studiekeuze volgend jaar.

Voor een eigenlijke implementatie van dit betaalsysteem zou DSpay nog verder ontwikkeld en getest moeten worden, zowel door gebruikers als ter beveiliging.

Of dit systeem ook echt in de praktijk wordt gebracht of niet, doet er niet echt toe. Als dit eindwerk maar als inspiratie kan dienen zodat toekomstige Schepperianen vlot en veilig elektronisch kunnen betalen.

Afbeeldingen

Afbeeldingen zonder bron zijn eigen afbeeldingen.

Bibliografie

Adafruit. (2016). *Adafruit 16x2 Character LCD + Keypad for Raspberry Pi.* Opgeroepen op oktober 9, 2016, van Adafruit: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-16x2-character-lcd-plus-keypad-for-raspberry-pi.pdf

Azat, M. (2015). Full Stack JavaScript: Learn Backbone.js, Node.js and MongoDB. Apress.

Computerphile. (2014, april 7). *The Tweeting Vending Machine Hack - Computerphile [Videobestand]*. Opgehaald van YouTube: https://youtu.be/-NVex5tVGy8

Cox, T. (2014). Raspberry Pi Cookbook for Python Programmers. Packt.

Donat, W. (2014). Learn Raspberry Pi Programming with Python. Apress.

FlowPilots. (sd). *Approach.* Opgeroepen op december 24, 2016, van FlowPilots: https://flowpilots.com/approach/

Hows, D., & Membrey, P. (2013). *Learn Raspberry Pi with Linux.* Apress.

ISO/IEC. (2013, maart 15). *Information technology — Radio frequency identification (RFID) for item management: Data protocol —.* Opgeroepen op december 25, 2016, van iso.org: https://www.iso.org/obp/ui/#iso:std:iso-iec:15961:-1:ed-1:v1:en

Jordi, S., Francesc, S., Vittorio, P., & M., N. (2016). *FingerScanner: Embedding a Fingerprint Scanner in a Raspberry Pi.* Opgehaald van MDPI Open Access Journals: http://www.mdpi.com/1424-8220/16/2/220/pdf

Krohn, O. (2014, december 22). Arduino-based Coffeemaker Payment System controls my Jura Impressa S95 [Videobestand]. Opgehaald van YouTube: https://youtu.be/uXPG3yZxTzM

Maklay, T. (2014, juli 20). *Simple NFC payment systems by Raspberry Pi and NFC PN532 [Videobestand]*. Opgehaald van YouTube: https://youtu.be/BYhclCBgLxM

OMahony, D., Peirce, M., & Tewari, H. (sd). *7 Electronic Payment Systems*. Opgeroepen op oktober 9, 2016, van https://www.medien.ifi.lmu.de/lehre/ws0910/mmn/mmn7.pdf

Pourghomi, P., & Ghinea, G. (2012). Managing NFC payment applications through cloud computing. IEEE.

Rajchel, L. (sd). *ISO/IEC JTC 1*. Opgeroepen op december 28, 2016, van http://www.iso.org/iso/jtc1_home.html

Roosa, S., & Schultze, S. (2010). The" Certificate Authority" trust model for SSL: a defective foundation for encrypted Web traffic and a legal quagmire. ProQuest.

RzTV. (2016, juli 25). *RFID/NFC Attendance & Payment System using Raspberry Pi [Videobestand]*. Opgehaald van YouTube: https://youtu.be/Xgf9mkEm77s

Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE.

Verklarende woordenlijst

Hieronder volgt een alfabetische lijst van hopelijk alle technische termen die ik doorheen dit eindwerk heb gebruikt. De definities die ik daarbij geef zijn uiteraard in de context van dit eindwerk bedoeld.

1. Actieve NFC: Vorm van NFC waarbij de NFC-tag een eigen stroombron heeft.
2. ADC: Analog to Digital Convertor, component dat signalen van analoge sensoren zoals bijvoorbeeld een potentiometer omzet in digitale signalen.
3. Airdrop: technologie door Apple voor het overzetten van bestanden via een combinatie van NFC en Bluetooth.
4. API: Application Interface: een manier om informatie van een systeem vanaf een externe aplicatie op te vragen of aan te passen.
5. Auth: Kort voor authentication.
6. Authentiseren: De identiteit van een gebruiker op een systeem verifiëren.
7. Back-end: Deel van een systeem dat de gebruiker niet te zien krijgt en de verwerking van gegevens op zich neemt.
8. Clientside: Zie front-end.
9. CPU: Central Processing Unit, belangrijkste component van microcontrollers en computers in het algemeen. Verwerkt berekeningen.
10. DAC: Digital to Analog Convertor, component in microcontrollers dat digitale signalen in een analoog signaal voor bijvoorbeeld geluid omzet.
11. Elektrische inductie: het genereren van elektrische energie door middel van straling.
12. Favicon: Icoontje van een webpagina of site.
13. Front-end: Deel van een systeem waarmee de gebruiker interageert.
14. Heartbeat: Systeem dat de connectie tussen twee apparaten controleert door om telkens na een tijdsinterval een bericht te sturen.
15. HID: Human Input Device, apparaat of elektronisch component dat een computer van informatie door een operator voorziet.
16. I2C: Inter-Integrated Circuit, communicatieprotocol specifiek bedoeld voor IC's zoals microcontrollers.
17. *ICE*: de 'International Electrotechnical Commission', opgericht in 1906, is een organisatie die standaarden op het vlak van elektrotechnologie ontwerpt en documenteert en nauw samenwerkt met de ISO.
18. ID: Identification, een code of nummer dat instaat voor de identiteit van een gebruiker.
19. Identificeren: De identiteit van een gebruiker aan een systeem bekendmaken.
20. Interpreter: Software die code interpreteert en converteert naar door een computer begrijpbare instructies.
21. IO: In-Out, afkorting voor elektrische verbinding op bijvoorbeeld een microcontroller om met andere apparaten te communiceren.
22. *ISO/IEC JTC 1*: de 'joint technical committee 1' van de ISO en de ICE, gevormd in 1987 is een comité dat zich inzet voor de standaardisatie van nieuwe technologieën op het vlak van ICT (informatie- en communicatietechnologie).
23. *ISO*: de 'International Organization for Standardization' ontwerpt en documenteert sinds 1947 wereldwijd industriele en commerciele standaarden.
24. LCD: Liquid-Crystal Display, displaytechnologie opgebouwd uit een speciale laag die licht al dan niet doorlaat of beïnvloed en een backlight om licht te voorzien.
25. LED: Light Emitting Diode, elektrisch component dat met weinig stroomverbruik licht uitstraalt.
26. MAC-adres: Media Access Control address, unieke code voor bijna alle communicatieapparaten apparaten.
27. *Master-slave*: een communicatiemodel waarbij de twee verbonden apparaten een onderscheiden rol vervullen. De slave kan alleen data verzenden wanneer de master daarvoor vraagt. Voorbeelden van technologien die dit model gebruiken zijn usb en NFC.

28. MDB: Multi Drop Bus, internationale standaard voor communicatie tussen betaalsystemen van automaten.
29. Microcontroller: Mini-computer voor simpele taken en berekeningen op één IC.
30. Multi-threaded: Computersysteem dat meerder taken tegelijk kan afhandelen.
31. NFC: Near Field Communication, toepassing van RFID-HF. Gebruikt voor dataoverdracht op korte afstand.
32. NFC-reader: Apparaat dat NFC-tags kan uitlezen en meestal ook aanpassen.
33. NIC: Network Interface Card, computeronderdeel via WiFi of ethernet verbinding maakt met een netwerk.
34. OLED: Organic Light Emitting Diode, type LED dat vooral in moderne displays wordt gebruikt, meestal helderder en efficiënter dan LCD's.
35. OS: Operating System, software dat andere programma's en services draait en beheert.
36. Passieve NFC: Vorm van NFC waarbij de NFC-tag voeding krijgt via de energie van radiogolven.
37. QR-code: Tweedimensionale code voor simpele visuele dataopslag.
38. Radio-transceiver: zie transceiver.
39. Relays: Elektrisch component om een ander circuit (vaak met een hoger vermogen) aan te sturen.
40. RFID: Radio Frequency Identification, technologie voor overdracht van kleine hoeveelheden data over radiogolven.
41. Serverside: Zie back-end.
42. SOC's: System On a Chip, Mini-computer met alle benodigde componenten op één chip.
43. SPI: Serial Peripheral Interface, master-slave communicatieprotocol, meestal gebruikt in LCD's, OLED's en microcontrollers.
44. Spoof: 'To spoof', de identiteit van een apparaat vervalsen.
45. Transceiver: een gecombineerde zender (transmitter) en ontvanger (receiver) van elektromagnetische golven.
46. Wireframe: Ontwerp met waarbij enkel de plaats van elementen is aangeduid.

Bijlagen

1. Interview met dhr. Vercauteren

Dit is het [geparafraseerde?] transscript van een interview dat ik afnam met dhr. Vercauteren op dinsdag 15 november 2016. Het luidere en van ruis verwijderde maar verder volledig originele audiobestand kan via deze link beluisterd worden: https://goo.gl/tMrMEv (of voluit: https://drive.google.com/open?id=0B6v6kb1Sbt-6ODlBeFg3VkVoejQ).

- Het eerste systeem dat de school gebruikte was de U-key?
    - Ja, dat bedrijf heette Xafax en het sleuteltje noemt een U-key.
- En Xafax was een Belgisch bedrijf?
    - Nee, dat was een Europees bedrijf maar nu vind je nog Xafax in Nederland.
- Want de U-key is eigenlijk Oostenrijks denk ik?
    - Ja, die sleutel is inderdaad Oostenrijks.
- En Xafax heeft die ingevoerd?
    - Ja zij voeren die in.
- En wanneer was dat dan?
    - Daar ga ik moeten achter zoeken. 2000 denk ik.
- En dan daarna, dat waren dan direct de laatste sleutels?
    - Ja, omdat Xafax te duur werd. Dan zijn we overgeschakeld. Dat was wel eenzelfde systeem, een betaalsysteem, waarmee je contactloos kon opladen. Maar het werkt volgens een ander principe. Xafax had een eigen protocol, die anderen hadden een universeler protocol. Ik weet niet meer juist de naam van dat protocol… Maar zo'n sleutel kon je dus gelijk waar kopen want dat is een vast protocol om contactloos te betalen.
- En welk bedrijf was da dan?
    - Die waren gevestigd in Sint-Katelijne-Waver maar het bedrijf bestaat niet meer.
- En wanneer was die overgang?
    - Da weet ik niet meer, wanneer dat is overgenomen en hoe dat da nu juist noemde, daar ga ik achter moeten zoeken.
    - Maar zij deden veel automatisatie, zoals die sleutels. Maar dan is da bedrijf overgenomen en hun beste werknemer is vertrokken en dan ja...
    - Het bedrijf heette Actips.
      En daarvoor Xafax met MIFARE.
- Dat is wat er in die NFC-tags zit?
    - Ja, en die hebben een bepaalde standaard.
- MIFARE daar zijn heel veel tags op gebaseerd.
    - Ja da's juist, en dat gebruikte die als standaard.
- Dus da kan zijn da Xafax ook MIFARE gebruikte?
    - Xafax is een verzamelnaam… dat is een firma, en die heeft onder anderen de U-key, dus die MIFARE-sleutel maar ook kaarten he. Hier da's een studentenkaart me een microchip in. Als je daar MIFARE insteekt gaat dat ook, zoals me de tag, da is zo een rondje dat BMW ook gebruikte. Dus ge kon die technologie eigenlijk steken waar je wilt.
    - En daar zit een antennetje in en in de eerste versie moest je dat er nog altijd in steken, in een toestel. Maar de latere versie moest je er gewoon tegenhouden. Dus de nieuwere systemen, het Actips systeem, daar kon je de sleutel er gewoon tegen houden.
    - En Actips was de firma en die zijn dan overgenomen.
    - Ja er zijn nog firma's, zoals Antenor die doen dat ook nog doen maar het grote probleem is die allemaal onderhoudscontracten op hun systemen willen.
- Ah, da's nog een ander ontwerp precies?

    - Ja, ESE payment systems, die hebben dat ook. Maar daar hebben we dan niet mee gewerkt. Die hebben Actips overgenomen en waren gevestigd in Affligem. Dus als je met die firma in zee gaat dan moet je nog betalen voor de reistijd dus ben je beter af dat je dat niet doet. Je hebt ook verschillen op het vlak dat Xafax aan elkaar hing, dat werd constant gemonitord. Er zijn verschillende systemen, er zijn er die zich constant monitoren als een soort apart netwerk dus op de computer zie je constant welke activiteit er gebeurt.
- Dus de betaalterminals zijn fysiek met elkaar verbonden?
    - Ja, maar je hebt ook systemen waar je de informatie kunt uitlezen, zodanig dat je die draden niet moet trekken. En dat is uiteraard vele interessanter want mij interesseert da eigenlijk niet hoeveel cola's dat je dronk, want dat kon je daar op zien he. Maar we hebben die vraag nooit gekregen, wij gebruiken dat uiteindelijk alleen om te betalen.
- Maar die waren dus verbonden met een draad en niet over wifi of zo?
    - Ja, da was met een draad maar dat was toen, dat was een van de eerste van zulke systemen. Maar dat betekende wel dat als er aan die draad een connectie fout liep dat heel het systeem platlag.
- Da's inderdaad wel een moeilijk systeem dan.
    - Ja, dat was wel tof maar daar hadden we heel veel nadeel aan. Maar dus dat Actips systeem daar stond de info in de kast en dan kon je dat inladen. Je maakt daar connectie mee met een computer en dan had je al de inhoud van da laadterminal. Dat was dus niet real-time maar wel gemakkelijk. Je kon daar dan ook mee een sleutel blokkeren zo dat als er iemand een sleutel steelt dat je gewoon kunt zeggen dat hij met die gestolen sleutel niets meer kon aanvangen.
- Ik had nog een ander vraagje; de studentekaarten, kun je daar een chip in stoppen?
    - Ja, wij maken die zelf dus dan kun je die kopen met een chip. En we hadden daar toen ook aan gedacht maar zo'n sleutel is voor een leerling plezanter en minder verliesbaar dan een kaart.
- Maar u studentenkaart moet je toch altijd hebben…
    - Inderdaad, dus we hebben daar ook aan getwijfeld want we hebben een tijd met kaarten gewerkt maar er is toen beslist dat voor de leerkrachten die die kaart willen houden die kunnen die houden en zij die een sleutel willen krijgen een sleutel. Een kaart is wel gevoeliger dan een sleutel, hoewel da eigenlijk hetzelfde is. Een sleutel is wat robuuster, je kunt die in een wasmachine stoppen en die werkt nog.
    - Dus dat gaat zeker, die kaarten printen. We kunne dat met een kaartdrukker, geen gesofisticeerde want we drukken de achterkant in het zwart maar de voorkant die drukken we in kleur. Dat werkt met linten die erover gaan met verschillende kleuren en dan maakt dat het beeld dat je wilt.
- En hoe werkt dat als je er een chip in wilt verwerken?
    - Dan is die kaart wat duurder, want jij moet die tag nog aan u systeem gaan activeren.
- Maar je kunt die kaarten gewoon met de chip erin kopen?
    - Ja, en dan heb je eigenlijk een systeem waarmee je informatie contactloos kunt overbrengen door het er gewoon op te leggen.
- En welk bedrijf maakt die kaarten?
    - Offini denk ik, maar dat moet je aan mevrouw Maes vragen, zij besteld die kaarten.
- En dan een laatste vraag; ik wou een enquête doen voor de ouders…
    - En dat is online?
- Ja, om dus te vragen of ze bv. zouden willen betalen via de schoolrekening. Maar het is denk ik toch beter om op voorhand te laten betalen.
    - Ja, pre-paid het voordeel daarvan is da da altijd betaald zal geraken.
- Inderdaad, maar het is eigenlijk gewoon een onderzoekje ze.
    - Oke want we hebben dat pre-paid systeem genomen omdat dat het goedkoopste systeem was. Maar er zijn ook systemen waar je kunt zeggen ik stort honderd euro op de rekening en

dan komt dat direct op dat machientje, maar dat kost stukken van mensen. Daarmee hebben we dat niet genomen maar eigenlijk is dat nog handig want nu staan we hier met geld en moeten we daarmee naar de bank dan moet je lobbyen met de bank maar die vragen geld voor alles dus dan zeg ik nee dat doe ik niet.

- Ja, inderdaad, maar daar zijn goede sytemen voor die met de meeste belgische banken werken.
    - Dan ben ik nieuwsgierig en zal ik dat dan lezen he.
- Oke da's goed, nu weet ik genoeg! Dankuwel.

2. Enquête aan de ouders

Dit is een enquête die ik gemaakt heb met Google forms die dankzij m. Van Langenhove via de Scheppers nieuwsbrief en op de schoolwebsite werd verspreid en met dank aan mijn moeder naar de leden van de ouderraad werd doorgemaild. De enquête bevat zowel vragen over het huidige betaalsysteem als over wat de ouders zien in een nieuw systeem. De enquête zoals de ouders die zagen is te vinden op: https://goo.gl/forms/XdH85HAOA5onur532

De vragen luidde als volgt:

A. Vind u dat het huidige systeem (contant betalen) vlot verloopt?
B. Indien neen: is dit zo omdat...
    a. u telkens geld moet meegeven aan uw zoon/dochter?
    b. u contant betalen onveilig vindt?
C. Zou u uw zoon/dochter liever laten betalen...
    a. Pre-paid met studentenkaart
    b. Pre-paid met sleutel
    c. Credit (via schoolrekening) met studentenkaart
    d. Credit (via schoolrekening) met sleutel
    e. Met bancontact
D. Zou u het voldoende veilig vinden dat enkel de studentenkaart of betaalsleutel nodig is voor een betaling?
E. Zou u akkoord gaan om biometrische identificatie (bv. vingerafdruk) te gebruiken bij betalingen?

De antwoorden en visualisaties vind u hier: https://goo.gl/JW7i3i

Naast de antwoorden op de vragen kreeg ik ook verscheidene suggesties voor mijn ontwerp:

"Het zou handig zijn dat de leerlingen behalve een broodje ook een drankje zouden kunnen kopen. Met electronisch (sic) betaalsysteem is dit voor de school geen extra moeite vind ik!"

"Bij mij op het werk hebben we een identiek system - onze identificatie badge (alsook badge waarmee deuren geopend kunnen worden) kan opgeladen worden zowel via BC kaart of cash. Hiermee betalen we 's middags in het bedrijfsrestaurant maar ook in de drank/snoep automaten."

"Het belangrijkste lijkt me een betaalmiddel te gebruiken waarbij bij verlies of diefstal het betaalmiddel waardeloos is voor externen.
En natuurlijk: succes met je eindwerk!!!"

"Voorkeur naar een bestaand betaalsysteem zoals bvb het contactloos betalen met een bankkaart."

"Ik vind de cash methode goed omdat onze zoon op deze manier een beter zicht heeft op wat hij allemaal uitgeeft."

"Goed initiatief deze enquête en succes met je eindwerk !"

"voorstel: betalingen via bancontact app op smartphone"