TECHNISCH ONTWERP

Energy Central

R. Troost

vrijdag 9 mei 2014

v0.3

Inhoudsopgave

1 Ontwikkelomgeving 3

2 Specificatie van de nieuwe interface 5

3 Beveiliging en onderhoud 9

4 Bronvermelding 10

# 1 Ontwikkelomgeving

***1.1 De technische infrastructuur***

De nieuwe applicatie vereist dat er een nieuwe server komt, welke nodig is om de data van verschillende apparaten te verzamelen. Aangezien er een Raspberry Pi gebruikt gaat worden voor de clients en er een Debian OS op gaat draaien, moet er zorgvuldig worden omgegaan met de schijfruimte en het geheugen van het apparaat. De clients en de centrale server hebben de volgende minimale specificaties nodig om de webserver efficiënt te kunnen draaien:

**Hardware (Server / client)**

* 1-2 core processor, 1.4 ghz+
* 15 GB+ Harde schijf
* 512MB+ RAM is aanbevolen

**Software**

* Mini Linux OS (Debian 7.0+ voor Raspberry Pi)
* Nginx Webserver met de volgende geïnstalleerde modules:
  + php-apc
  + php5-fpm
  + php5-cli
  + php5-gd
  + php5-mcrypt
  + php5-mysql
  + php5-curl
  + php5-intl
* MySQL server

***1.2 Inrichten van de ontwikkelomgeving***

Om de applicatie te testen en te ontwikkelen, zal er gebruik gemaakt worden van een ontwikkelomgeving. Deze zal worden opgezet met een Debian OS door middel van *Vagrant*. Deze tool kan door gebruik van een configuratie bestand de virtuele machine zelf aanmaken en installeren. Dit betekent dat als iemand anders mee wilt werken aan het project, hij dezelfde ontwikkelomgeving krijgt.

Het doel is dan ook om de productieomgeving zo goed mogelijk na te maken. Belangrijk is om de genoemde specificaties bij het kopje ‘technische infrastructuur’ zo goed mogelijk te volgen zodat er een duidelijk beeld ontstaat van hoe de applicatie uiteindelijk zal reageren in een productieomgeving.

***1.2.1 Clients en server (provisioning)***

De clients (apparaten) zullen worden ingericht met een tool genaamd *Ansible*. Ansible maakt het mogelijk om bepaalde scripts tegen een externe machine te draaien. Deze scripts worden 'plays' genoemd. Deze plays worden in een *playbook* opgenomen. Een play bestaat uit meerdere taken, bijvoorbeeld het installeren van een webserver.

Tijdens het installeren van de clients zal er een cron-job (taak) worden aangemaakt op de externe machine voor het uitrollen van het project. Deze cron-job zal elke twee uur checken of er een nieuwe versie van het project beschikbaar is en deze installeren.

De centrale server zal op dezelfde manier ingericht worden. Doordat een 'play' meestal idempotent is, is het mogelijk om het script meerdere malen te draaien. Nadat het script de eerste keer heeft gedraaid, is het dus mogelijk om deze een aantal weken later nog een keer te draaien. Een groot voordeel hiervan is dat bijvoorbeeld de software naar de laatste versie wordt bijgewerkt als dat nodig is.

De folder met de CSV bestanden op de centrale server zal ge-*CHMOD* worden naar 644 (-rw-r--r--) of minder rechten.

***1.2.2 Client (deployment)***

De clients kunnen het project dat op de webserver staat zelf updaten. Dit zal gebeuren door middel van het ansible-pull commando. Dit commando zal een GitHub repository binnenhalen op de machine waarop het commando wordt gedraaid. In dit geval zal dat de repository zijn waarin het deployment script staat. Dit zal ervoor zorgen dat altijd het laatste deployment script wordt gebruikt.

Het deployment script is een playbook met een play erin, welke taken bevat om de laatste versie van het project binnen te halen, en indien nodig, deze te installeren. Ook zal deze play ervoor zorgen dat er een cron-job wordt aangemaakt welke de CSV bestanden met data van de energieopbrengst erin elke 10 minuten naar de centrale server toe zal sturen.

***1.2.3 Programmeertaal en ontwikkeltools***

De website zal ontwikkeld worden door middel van een PHP Framework genaamd Silex. Silex is een Micro-framework welke standaard al een aantal componenten bevat. Naast Silex zullen er nog een aantal extra plugins gebruikt worden om het ontwikkelen van de website gemakkelijker te maken. De onderdelen van het Framework en de plugins die gebruikt zullen worden staan hieronder in een overzicht.

**Silex Micro-framework onderdelen**

* Twig (PHP template engine)
* Doctrine
  + DBAL (DataBase Abstraction Layer)
  + QueryBuilder
* Symfony 2

**Plugins / extras**

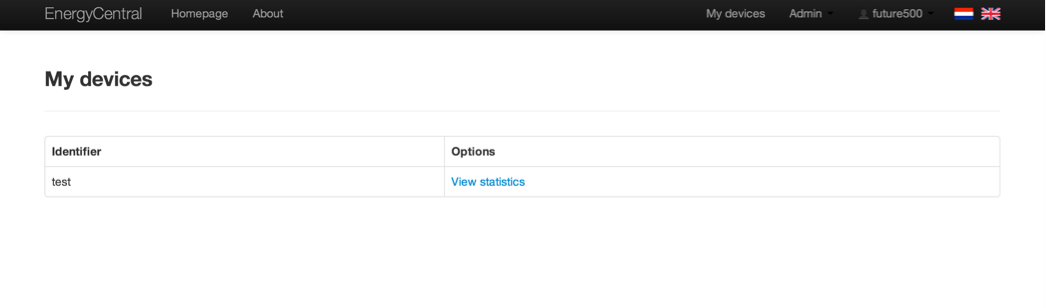
* jQuery v1.9+
* Highcharts (JavaScript library voor grafieken)
* Bootstrap v2.3.2
* Select2 (tagging)
* Silex pagination
* Date picker
* Ladda-spinner

De volgende tools zullen gebruikt worden tijdens het maken van de website en het inrichten van de ontwikkelomgeving:

* PhpStorm IDE
* Vagrant
* Ansible
* VirtualBox

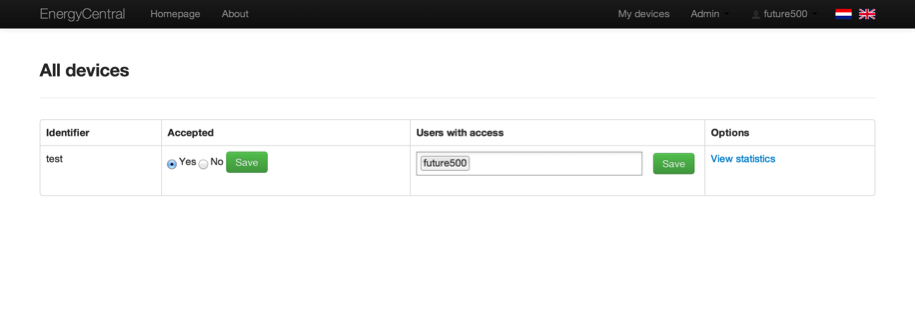
# 2 Specificatie van de nieuwe interface

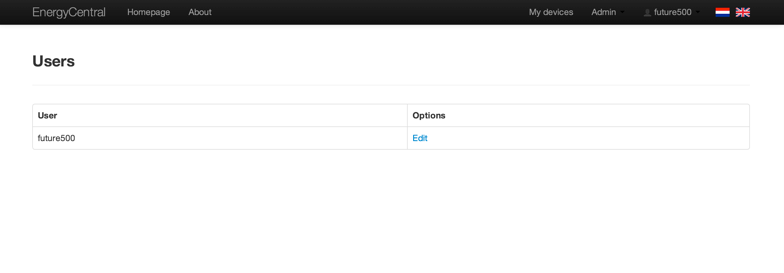
Een deel van de interface bestaat al, namelijk het bekijken van de statistieken van één apparaat. Deze functionaliteit zal worden meegenomen tijdens de uitbreiding van de applicatie. Er kan vanuit gegaan worden dat er jQuery wordt gebruikt om de taal te wijzigen via de taal icoontjes rechts bovenin. jQuery zal ook gebruikt worden om de data voor de grafieken te verkrijgen en daarna te tonen in een grafiek met de Highcharts library. Hieronder zal een technische beschrijving worden gegeven en zal door middel van afbeeldingen duidelijk worden gemaakt hoe bepaalde onderdelen gebouwd moeten worden.

**2.1 SQL queries**  
Het is handig om de *Doctrine DBAL Query Builder* te gebruiken omdat er conditionele queries zullen ontstaan. Als de queries handmatig aan elkaar 'geplakt' worden zal er sneller onduidelijke code ontstaan. Het doel van de Query Builder in dit project is het behouden van overzichtelijke code en het makkelijker schrijven van de SQL queries.  
  
**2.2 Beschrijving interfaces**  
  
Afbeelding 0.1: Apparaat overzicht (user), central mode only  
  
Op afbeelding 0.1 is te zien dat elke gebruiker zijn eigen apparaten kan zien in een lijst. De apparaten zullen worden opgehaald uit een toegangstabel waar de userid gelijk is aan de ingelogde user zijn userid. Ook wordt het bijbehorende ID van een device getoond in deze lijst. Om deze informatie te verkrijgen, zal er een join query gedaan moeten worden.

Voorbeeld query: ***SELECT dev.\*   
 FROM devaccess ac  
 INNER JOIN device dev ON ac.deviceid = dev.deviceid  
 WHERE userid = :userid***

De controller zal ervoor zorgen dat de data naar de template wordt toegestuurd. In de template zal er een 'for each' loop gedaan worden over de nummers, waarna deze in een tabel worden weergegeven (zie afbeelding 0.1).

  
Afbeelding 0.2: Apparaat overzicht (admin), central mode only



Afbeelding 0.3: Lijst met alle gebruikers (admin), central mode only

Op afbeelding 0.2 is te zien dat een beheerder alle bestaande apparaten kan bekijken. Op deze pagina kan toegang tot een apparaat worden verleend aan een gebruiker. Er zal een vast aantal apparaten worden getoond. Als er meer apparaten zijn dan dit aantal, zullen er onderin het beeld buttons verschijnen met pagina nummers.

Voor de *pagination* moet dus een nieuwe instantie van ‘PaginationServiceProvider’ geregistreerd worden. De buttons met pagina nummers worden opgebouwd d.m.v. het huidige pagina nummer, de hoeveelheid records en hoeveel records er per pagina getoond mogen worden.

Onder de "users with access" kolom zal gebruik worden gemaakt van de Select2 plugin.

Dit maakt het mogelijk om te gebruikers te "taggen". Voordat gebruikers getagd kunnen worden, moeten deze eerst worden geladen. Dit kan door middel van de volgende JavaScript code:

*$('.device').select2({*

*createSearchChoice: function() {   
 return null;*

*}, // Disable adding of non-existent elements*

*tags: {{ users|raw }},*

*width: '80%'*

*});*

De 'users' variabel zal door de controller naar de template worden gestuurd.

Zodra er op de 'save' knop wordt gedrukt, zal jQuery een POST request naar de controller maken met de nieuwe users. De controller verwerkt de request en zorgt dat de lijst met gebruikers wordt opgeslagen.  
  
Voorbeeld van een functie (in dit geval een ‘service’) om de toegang van gebruikers voor apparaten te regelen:

*$app['devices.update'] = $app->protect(*

*function ($userId, array $addedDevices, array $removedDevices) use ($app) {*

*$queryBuilder = $app['db']->createQueryBuilder();*

*foreach ($addedDevices as $deviceId) { // add new devices*

*// insert new devices..*

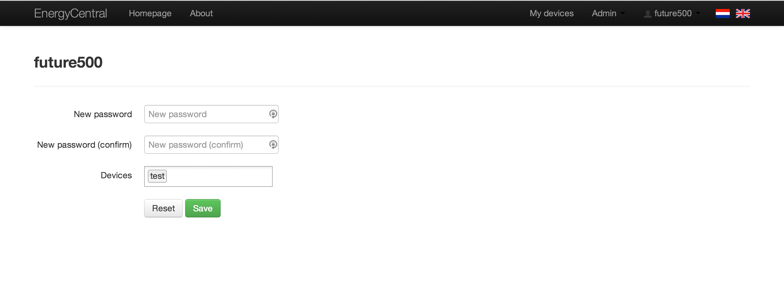
*}*

*foreach ($removedDevices as $deviceId) { // remove old devices if needed*

*// delete removed devices…*

*}*

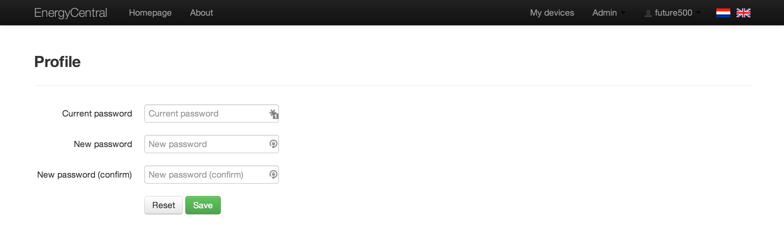
*);*



Afbeelding 0.4: Gebruiker wijzigen (admin), central mode only

In afbeelding 0.3 is te zien hoe een beheerder een gebruiker zijn details kan aanpassen. Op dit scherm kan een nieuw wachtwoord worden ingevoerd of nieuwe apparaten worden toegewezen aan een gebruiker. Het wachtwoord zal gesalt worden door middel van een random byte string gemaakt door de SecureRandom class van Symfony2. Deze byte string zal base64-encoded worden om te zorgen dat deze correct kan worden opgeslagen. Voorbeeld code om een salt te maken voor de wachtwoorden:

*$randGenerator = new SecureRandom();*

*base64\_encode($randGenerator->nextBytes($length));*  
  
Afbeelding 0.5: Profiel wijzigen (user), central mode only

***Lokale mode en centrale mode***

Wanneer er ingelogd is als normale gebruiker (ROLE\_USER), kunnen alleen de statistieken van het eigen apparaatje bekeken worden (net zoals in de lokale mode, maar dan op een centrale plek).

Voordat de statistieken van een apparaat worden getoond, wordt in de Twig template eerst gecheckt of de gebruiker toegang heeft tot het apparaat. Beheerders mogen alle apparaten bekijken. Dit kan gedaan worden door de volgende code:  
  
*{% if (device\_access or not app.centralmode) or is\_granted('ROLE\_ADMIN') %}*

*<< code >>*

*{% endif %}*

***Aandachtspunten realisatie***  
Een aantal zaken zullen van belang zijn tijdens de realisatie van de nieuwe functionaliteiten:

* Bestanden moeten vanaf de client (het apparaatje) worden verzonden als een CSV via een versleutelde verbinding.
  + Het versturen van dit bestand zal gedaan worden d.m.v. het ‘scp’ (securecopy) commando vanaf de client kant. De CSV zal dan op de server opgeslagen worden in een map met het formaat: \ec-<id>-dd\_mm\_yyyy.csv.

De volgende methodes om een bestand over te zetten of data naar een andere server te sturen zijn overwogen:

* FTP
* SFTP
* TCP/UDP (directe data stream)
* SCP (Secure copy commando)
* Rsync (incrementele bestandssynchronisatie)
* Directe MySQL verbinding (config bestand op het apparaat waarin de gegevens van de externe server staan)
* Data versturen naar een extern script (bijv. een PHP pagina) d.m.v. SSL

Er is vooral gelet op: de robuustheid, veiligheid en gemak van het gebruik.

* De identificatiecode van elk apparaat zal door het import script uit de bestandsnaam worden gehaald.
* Er moet een cron-job aangemaakt worden die elke 24 uur de CSV bestanden verstuurd.
  + Commando’s die mislukken tijdens het uitvoeren van de cron-job moeten worden gelogd.
  + Data die niet verstuurd wordt (bijv. door het uitvallen van internet, etc.) moet de eerst volgende keer meegestuurd worden.
* Cron-job task voor het versturen van de data:

*@daily of* 0 0 \* \* \* */home/<naam>/ec\_senddata.sh*

* De data zal eerst op het apparaat zelf geïmporteerd worden voordat deze verstuurd wordt. Dit wordt gedaan door het volgende commando: “  
    
  *<install dir>console import:run && <install dir>/console copy:run <user> <host>*
* Er komt een in/uitlog pagina, de toegang tot verschillende pagina’s zal geregeld worden door de een ACL (Access Control List)

Voordat de CSV bestanden gekopieerd kunnen worden naar de centrale server, moet de public key van de client worden toegevoegd aan de authorized\_keys op de server. We zullen de public/private key van de client vooraf genereren met het commando "*ssh-keygen -t rsa*" en de public key toevoegen op de server. Daarna kan een apparaat data versturen, maar voordat deze data wordt toegevoegd moet het apparaat eerst geaccepteerd worden door een beheerder.

# 3 Beveiliging en onderhoud

***3.1 Beveiliging***

Verschillende delen van de website zullen beveiligd worden d.m.v. een ACL (Access Control List). De Symfony firewall zal checken of de gebruiker is ingelogd. Daarna worden de regels van de ACL toegepast.

Voor de database zal er gebruik gemaakt worden van een gebruiker met de minste rechten. Dit is om te voorkomen dat er met de database structuur wordt geknoeid of dat er andere ongewenste dingen gebeuren.

De wachtwoorden in de database zullen gesalt worden om te voorkomen dat deze gemakkelijk gekraakt kunnen worden. De volgende code kan gebruikt worden om een salt te genereren:

*$app['user.generatesalt'] = $app->protect(*

*function ($length = 25) use ($app) {*

*$randGenerator = new SecureRandom();*

*return base64\_encode($randGenerator->nextBytes($length));*

*}*

*);*

***3.2 Beheer***

De CSV bestanden zullen eerst op het apparaat zelf geïmporteerd worden. Zodra dit is gedaan, wordt het bestand naar de ´centrale´ server gestuurd. Als deze het bestand ook met succes importeert dan kan het apparaat de bestanden weggooien. Er is verteld dat er niet heel veel mensen zijn die de applicatie zullen gaan gebruiken. De verwachting is dan ook dat de database gewoon kan groeien in grootte en er geen onderhoud verricht hoeft te worden.

# 4 Bronvermelding

De volgende bronnen zijn gebruikt tijdens het maken van dit technisch ontwerp:

* Functioneel ontwerp *(vragen en informatie over functionaliteiten)*