TECHNISCH ONTWERP

Energy Central

R. Troost

woensdag 21 mei 2014

v0.4.0

**Inhoudsopgave**

1 Samenvatting 3

1.1 Samenvatting voor de klant 3

1.2 Versie overzicht 3

2 Ontwikkelomgeving 4

2.1 De technische infrastructuur 4

2.2 Inrichten van de ontwikkelomgeving 4

2.2.1 Clients en server (*provisioning*) 4

2.2.2 Client (*deployment*) 5

2.2.3 Programmeertaal en ontwikkeltools 5

3 Specificatie van de nieuwe interface 6

4 Specificatie van de database 15

4.1 Specificatie van de database 15

5 Beveiliging en onderhoud 16

5.1 Beveiliging 16

5.2 Beheer 16

6 Termen 16

7 Bronvermelding 17

# 1 Samenvatting

## 1.1 Samenvatting voor de klant

Dit document is geschreven om technisch inzicht te geven over hoe de nieuwe functionaliteiten in het Energy Central project gerealiseerd moeten worden. De uitbreiding van de applicatie zal een aantal nieuwe *dependencies* met zich meebrengen, zoals: de Silex pagination, Ladda-spinner en de DateJS library. Er moet een centrale server ingericht worden met een aantal PHP-extensies en de database moet opgezet worden.

## 1.2 Versie overzicht

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versie** | **Status** | **Datum** | **Auteur** | **Wijzigingen** |
| 0.1.0 | Incompleet | 27/02/14 | R. Troost | Eerste versie. |
| 0.2.0 | Incompleet | 05/03/14 | R. Troost | - Screenshots bijgewerkt.  - Beschrijving ontwikkelomgeving bijgewerkt.  - Beschrijving beveiliging en onderhoud bijgewerkt. |
| 0.3.0 | Incompleet | 19/05/14 | R. Troost | - Samenvatting toegevoegd.  - Lijst met termen toegevoegd. |
| 0.3.1 | Incompleet | 20/05/14 | R. Troost | - Screenshots en beschrijvingen bijgewerkt.  - Specificatie van de database toegevoegd. |
| 0.4.0 | Incompleet | 21/05/14 | R. Troost | - Sections opnieuw ingedeeld.  - Beveiliging en onderhoud bijgewerkt.  - Samenvatting voor de klant bijgewerkt.  - Inhoudsopgave bijgewerkt. |

# 2 Ontwikkelomgeving

## 2.1 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 welke op een Debian OS 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 met de geïmporteerde *skeleton* *database*

## 2.2 Inrichten van de ontwikkelomgeving

Om de applicatie te testen en te ontwikkelen, zal er gebruik gemaakt worden van een virtuele 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 veel mogelijk te volgen zodat er een duidelijk beeld ontstaat van hoe de applicatie uiteindelijk zal reageren in een productieomgeving.

## 2.2.1 Clients en server (*provisioning*)

De clients (apparaten) zullen worden ingericht door 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 kan bestaan uit één of meerdere taken, bijvoorbeeld het installeren van een webserver.

Tijdens het installeren van de clients zal er een *cron-job* 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.

## 2.2.2 Client (*deployment*)

De clients kunnen het project dat op de server 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.

## 2.2.3 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
  + QueryBuilder
* Symfony 2

**Plugins / extras**

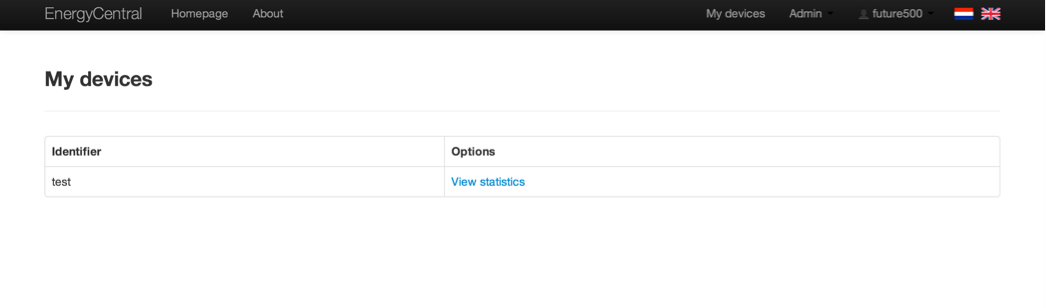
* jQuery v1.9+
* Highcharts (JavaScript library voor grafieken)
* Bootstrap v2.3.2
* Select2 (tagging)
* Silex pagination
* Date picker
* Ladda-spinner
* DateJS library

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

* PhpStorm IDE
* Sublime Text 2
* Vagrant
* Ansible
* VirtualBox

# 3 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.

3.1 SQL query’s  
Het is handig om de Doctrine DBAL Query Builder te gebruiken omdat er conditionele query’s zullen ontstaan. Als de query’s 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 query’s.  
  
3.2 Beschrijving interfaces  
**Afbeelding 0.1: Apparaat overzicht (user), central mode only**

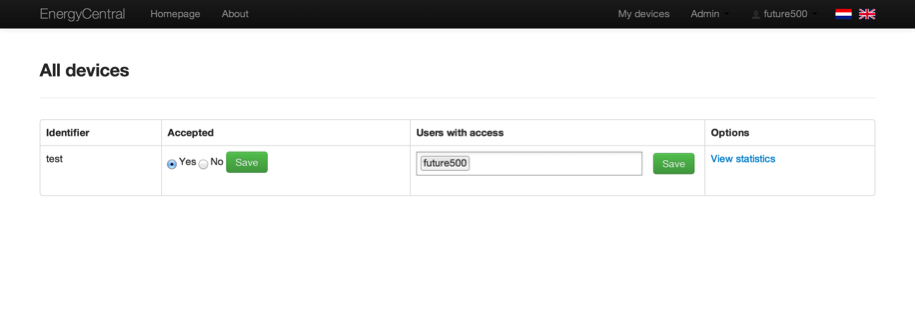
***Extra benodigdheden / eisen***

- De apparaten moeten worden opgehaald uit een toegangstabel.  
- Alleen de apparaten die toegewezen zijn aan de huidige user ID worden opgehaald inclusief de identifier.   
  
  
  
***Beschrijving interface***  
In afbeelding 0.1 wordt getoond dat een gebruiker alle apparaten die aan hem toegewezen zijn kan zien in een lijst.

Om deze informatie te verkrijgen, zal er een join query gedaan moeten worden.  
  
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.  
  
***Voorbeelden***

Apparaten waar een bepaalde gebruiker toegang toe heeft ophalen:  
*SELECT dev.\*   
FROM devaccess ac*

*INNER JOIN device dev ON ac.deviceid = dev.deviceid  
WHERE userid = :userid*

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

***Extra benodigdheden / eisen***  
- Instantie van PaginationServiceProvider

- Pagination buttons onderin als het aantal items meer dan een vooraf gedefinieerd aantal is.  
*Deze wordt opgebouwd d.m.v. het huidige pagina nummer, de hoeveelheid records en hoeveel records er per pagina getoond mogen worden.*  
- Select2 plugin

- Lada-spinner ***Beschrijving interface***   
In 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.

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.

De 'users' variabele 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 ervoor dat de toegangslijst voor het apparaat wordt opgeslagen.  
  
Om aan te tonen dat er een request gedaan wordt, zal de tekst van de knop veranderen in een spinner. Dit kan gedaan worden door middel van de lada-spinner library.

***Voorbeelden***

Service die de toegang tot apparaten bijwerkt voor een bepaalde gebruiker:

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

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

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

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

*// ...*

*}*

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

*// ...*

*}*

*);*

Taggen van users met Select2:

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

*createSearchChoice: function() {   
 return null; // Disable adding of non-existent elements*

*},*

*tags: {{ loaded\_users }},*

*width: '80%'*

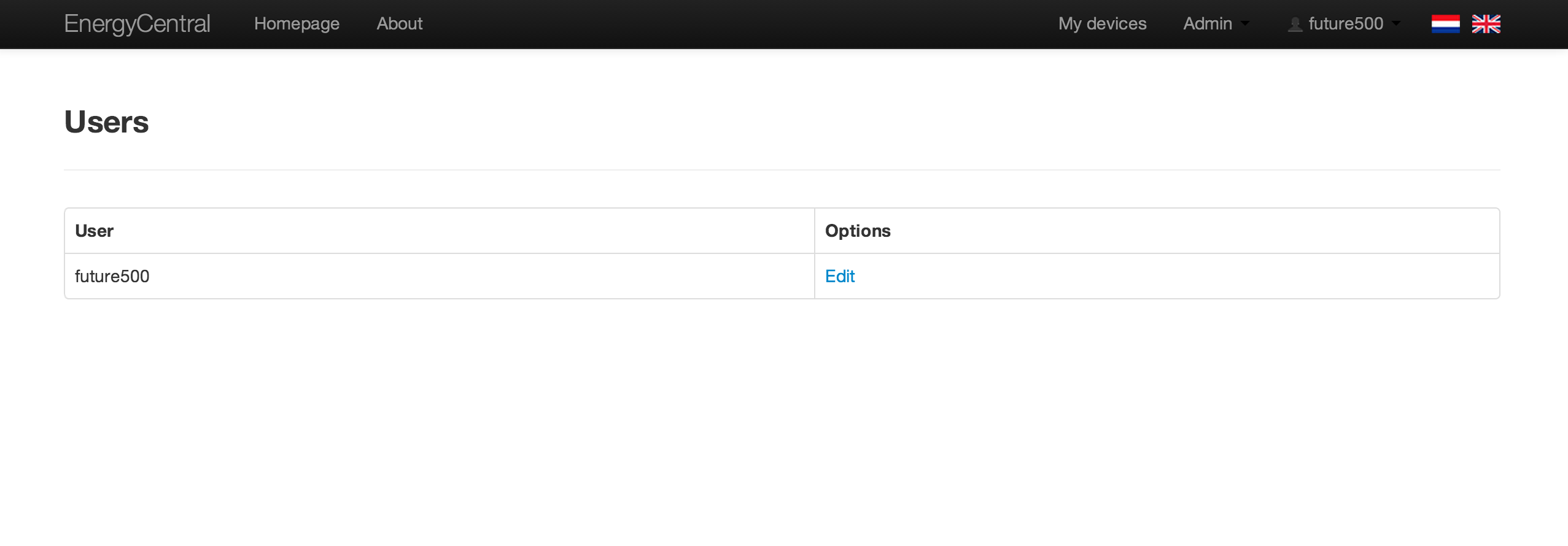
*});*

Lada-spinner voorbeeld (JavaScript):  
  
 *function updateAccess(deviceId)*

*{*

*var spinner = Ladda.create( document.querySelector( '#dev' + deviceId + '\_save' ) );*

*spinner.start();  
  
 // ....  
  
 spinner.stop();  
 }*



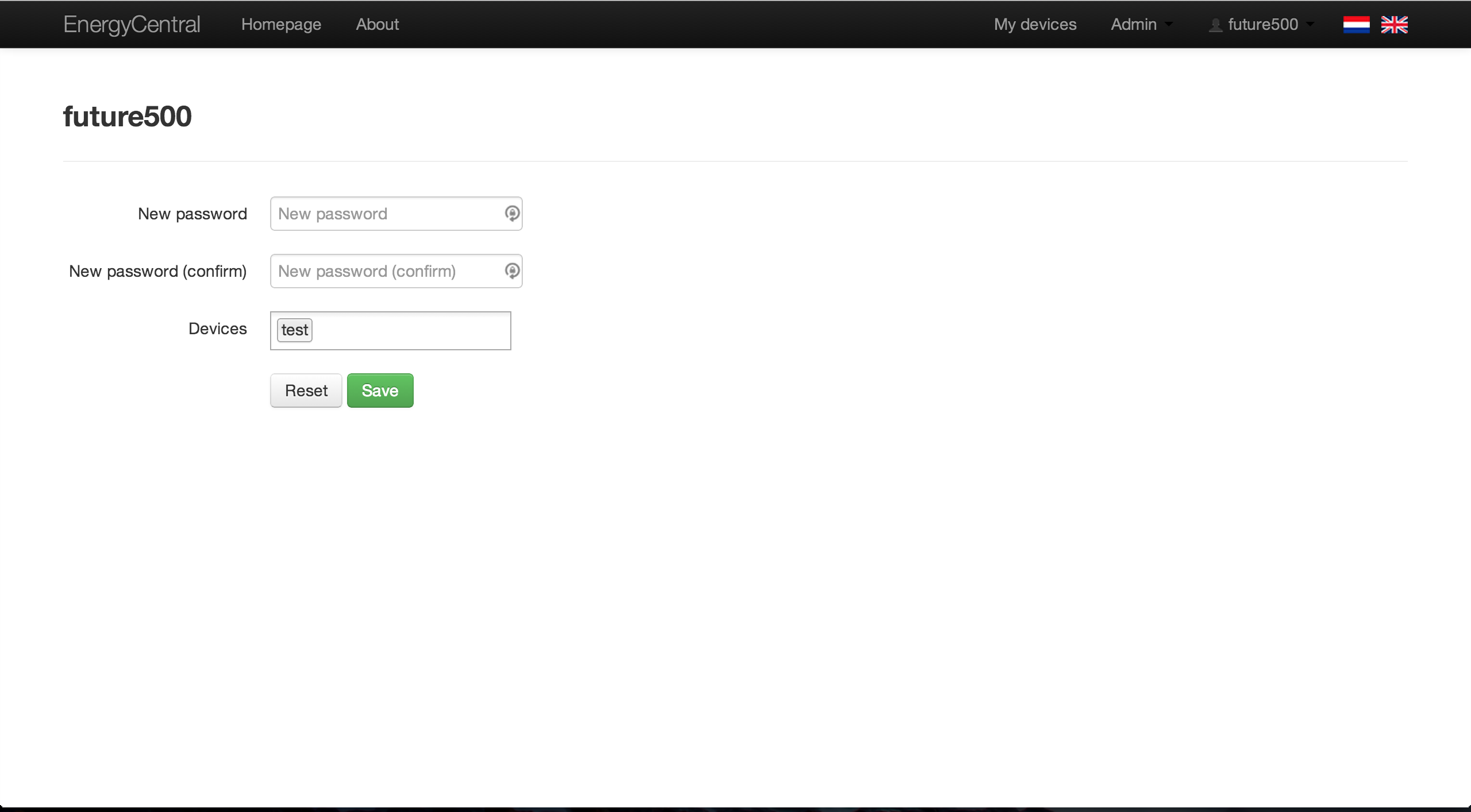
**Afbeelding 0.3: Lijst met alle gebruikers (admin), central mode only**

***Extra benodigdheden / eisen***  
- Instantie van PaginationServiceProvider

- Pagination buttons onderin als het aantal items meer dan een vooraf gedefinieerd aantal is.  
*Deze wordt opgebouwd d.m.v. het huidige pagina nummer, de hoeveelheid records en hoeveel records er per pagina getoond mogen worden.*

***Beschrijving interface***  
Een beheerder kan een lijst met alle gebruikers zien (afbeelding 0.3). Ook hier geldt dat er pagination wordt gebruikt. De lijst met gebruikers wordt opgehaald uit de **user** tabel. 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 gebruikers, waarna deze in een tabel worden weergegeven.  
  
***Voorbeelden***Gebruikers ophalen uit de user tabel:  
  
*SELECT u.username (of " \* ")  
FROM user u*

*LIMIT x, y*

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

***Extra benodigdheden / eisen***

- Select2 plugin   
  
- Het wachtwoord moet gesalt worden door middel van een random byte string gemaakt door de SecureRandom class van Symfony2. Deze byte string moet base64-encoded worden om te zorgen dat deze correct kan worden opgeslagen.

- De twee nieuwe wachtwoorden moeten gelijk aan elkaar zijn.  
- De twee nieuwe wachtwoorden mogen niet leeg zijn.

- Om het invullen van de apparaten namen gemakkelijker te maken zal de Select2 plugin gebruikt worden. Deze plugin maakt het mogelijk om de namen van apparaten te taggen.

***Beschrijving interface***

In afbeelding 0.4 is te zien hoe een beheerder het account van een gebruiker kan aanpassen. Op deze pagina kan een nieuw wachtwoord worden ingesteld of nieuwe apparaten worden toegewezen aan een gebruiker.

Zodra er op 'Save' wordt gedrukt, wordt er door de controller gecontroleerd of de nieuwe wachtwoorden gelijk aan elkaar zijn. Dit kan gedaan worden door gebruik te maken van de Symfony 2 validation constraints.

Als er errors zijn tijdens de validatie, worden deze getoond in de view. Ook wordt de apparaat toegangslijst bijgewerkt met eventueel nieuw toegevoegde (of verwijderde) apparaten voor de geselecteerde gebruiker.

***Voorbeelden***

Voorbeeld code om een salt te maken voor de wachtwoorden:

*$randGenerator = new SecureRandom();*

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

Valideren of de bevestiging van het nieuwe wachtwoord gelijk is aan het nieuwe wachtwoord:  
  
 *$constraint = new Assert\Collection(*

*array(*

*// ...*

*'new\_password\_confirm' => array(*

*new Assert\NotBlank(),*

*new Assert\Length(*

*array('max' => 100)*

*),*

*new Assert\EqualTo(*

*array(*

*'value' => $request->get('new\_password'), // must match new password*

*'message' => 'The new password does not match the confirmation password!'*

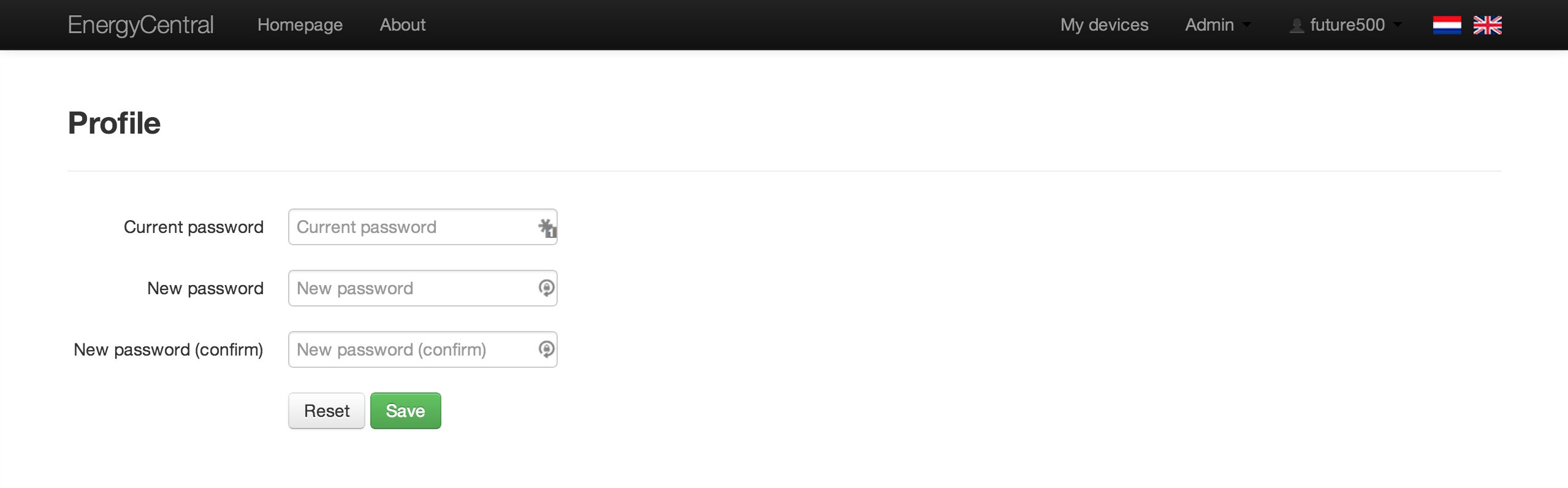
*)*

*)*

*)*

*)*

*);*



**Afbeelding 0.5: Profiel wijzigen (user), central mode only**

***Extra benodigdheden / eisen***

- Het wachtwoord moet gesalt worden door middel van een random byte string gemaakt door de SecureRandom class van Symfony2. Deze byte string moet base64-encoded worden om te zorgen dat deze correct kan worden opgeslagen.

- Er moet gecheckt worden of het ingevulde wachtwoord gelijk is aan het huidige wachtwoord.

- Er moet gecheckt worden of het nieuwe wachtwoord gelijk is aan het oude wachtwoord.

- Geen van de velden mag leeg zijn

***Beschrijving interface***

Een gebruiker heeft de mogelijkheid om zijn eigen profiel te bewerken. Wanneer er op "save'' wordt gedrukt, worden de wachtwoord velden gevalideerd door de controller.

Eventuele validatie errors worden door de controller naar de view gestuurd en daar getoond.

***Voorbeelden***

Voorbeeld code om het wachtwoord van de huidige gebruiker te valideren (deze gaat in een Assert\Collection als er meerdere validaties zijn):  
 *$constraint = new Assert\Collection(*

*array(*

*// ....*

*'old\_password' => array(*

*new Assert\NotBlank(),*

*new* ***SecurityAssert\UserPassword****(*

*array(*

*'message' => 'The old password you entered does not match!'*

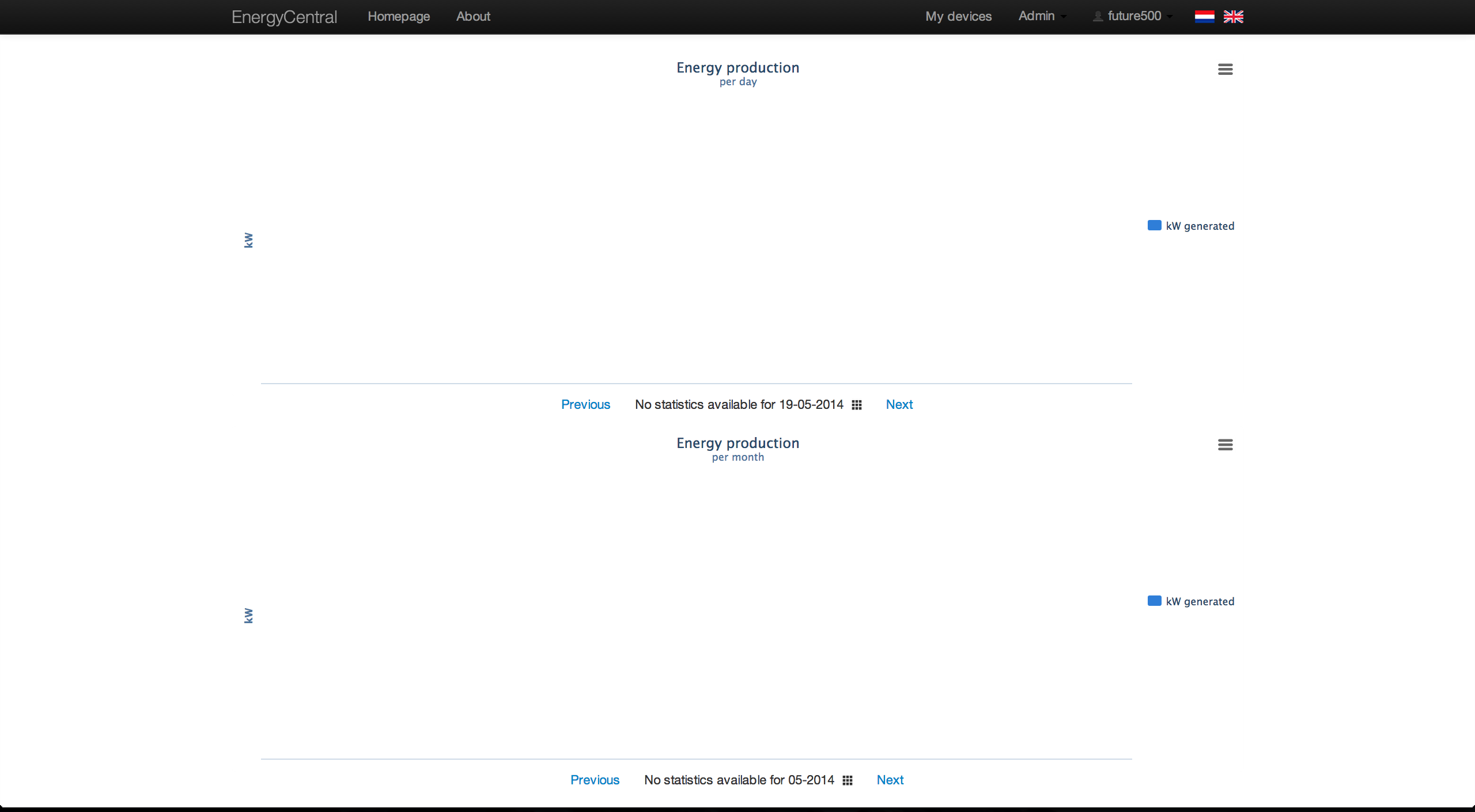
*)*

*)*

*),*

*)*

*)*

  
**Afbeelding 0.6 \*: Statistieken van een apparaat bekijken**\* = **functionaliteit bestaat al** ***Extra benodigdheden / eisen***

- Date picker  
- Highcharts library  
- DateJS library

- Er moet gecheckt worden of de previous / next link al de minimum / maximum datum is van wanneer er data beschikbaar is. Als dit zo is, dan wordt de link van de vorige/volgende text een '#'.

***Beschrijving interface***

Nadat er een apparaat gekozen is om de statistieken van te bekijken, wordt er een nieuwe pagina met twee grafieken getoond (zie afbeelding 0.6).  
  
- De grafieken worden getoond door middel van de Highcharts library.  
- De data voor de grafieken wordt door de controller naar de view gestuurd.  
- De 'previous' en 'next' link worden via JavaScript gezet. De datum kan berekend worden door +1 of -1 dag op de geselecteerde datum toe te passen.

- De datum wanneer er voor het eerst en voor het laatst data voorkomt wordt door de controller naar de view gestuurd.

***Voorbeelden***

Datum van de previous / next link veranderen:

*var prevDate = new Date.parse(selectedDate).add(-1).months();*

*var nextDate = new Date.parse(selectedDate).add(1).months();*

*$("#prev-month").attr("href", isMinDate ? '#' : "javascript:getMonth('" + prevDate.toString('yyyy-MM') + "')");*

*$("#next-month").attr("href", isMaxDate ? '#' : "javascript:getMonth('" + nextDate.toString('yyyy-MM') + "')");*

Checken of de huidige datum gelijk is aan de minimum / maximum datum (link wordt eventueel vervangen met een #)  
  
 *var isMinDate = new Date.parse(selectedDate).equals(Date.parse('{{ month.min }}'));*

*var isMaxDate = new Date.parse(selectedDate).equals(Date.parse('{{ month.max }}'));*

Datepicker:  
  
  *$("#icon-monthly").datepicker({*

*format: "yyyy-mm",*

*viewMode: "months",*

*minViewMode: "months",*

*autoclose: true*

*}).on('changeDate', function (e) {*

*getMonth(e.date.toString('yyyy-MM'));*

*});*

***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 (Raspberry PI) worden verzonden als een CSV via een versleutelde verbinding.
  + Het versturen van dit bestand zal gedaan worden d.m.v. het ‘sftp’ (**s**ecure**ftp**) 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. HTTPS

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.

# 4 Specificatie van de database

## 4.1 Specificatie van de database

Om de database structuur in kaart te brengen, worden hieronder de tabellen met hun bijbehorende attributen en types weergegeven.

**daydata**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| datetime | datetime | NO | PRI |  |  |
| deviceid | int(5) | NO | MUL |  |  |
| kWh | decimal(12,3) | YES |  | NULL |  |
| kW | decimal(12,3) | YES |  | NULL |  |

**devaccess**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| deviceid | int(5) | NO | PRI |  | auto\_increment |
| userid | int(4) | NO | PRI |  |  |

**device**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| deviceid | int(5) | NO | PRI |  | auto\_increment |
| name | varchar(25) | NO |  |  |  |
| accepted | tinyint(1) | NO |  |  |  |

**monthdata**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| date | date | NO | PRI |  |  |
| deviceid | int(5) | NO | MUL |  |  |
| kWh | decimal(12,3) | YES |  | NULL |  |
| kW | decimal(12,3) | YES |  | NULL |  |

**user**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| userid | int(4) | NO | PRI |  | auto\_increment |
| username | varchar(30) | NO |  |  |  |
| password | varchar(100) | NO |  |  |  |
| salt | varchar(255) | NO |  |  |  |
| roles | varchar(20) | NO |  |  |  |

# 5 Beveiliging en onderhoud

## 5.1 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. Om een gebruiker met alleen de rechten 'SELECT', 'INSERT', 'UPDATE' en 'DELETE' aan te maken kan de volgende SQL query gebruikt worden:  
  
 *CREATE USER*

*'test123'@'%' IDENTIFIED BY '<password>';*

*GRANT*

*SELECT, INSERT, UPDATE, DELETE*

*ON future500.\**

*TO 'test123'@'%';*

*FLUSH PRIVILEGES;*

De wachtwoorden in de database zullen gesalt worden om te voorkomen dat deze gemakkelijk gekraakt kunnen worden.

## 5.2 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.

# 6 Termen

|  |  |
| --- | --- |
| Term | Definitie |
| Ansible | Programma om het inrichten van een machine mee te automatiseren. |
| Cron-job | Geplande taak die met een bepaalde interval uitgevoerd kan worden. |
| CHMOD | **Ch**ange **mod**e, hiermee kunnen de bestands of folder permissies veranderd worden. |
| DBAL | DataBase Abstraction Layer |
| Dependency | Eén van de benodigdheden voor het project, bijvoorbeeld een aparte library. |
| Deploy | Het uitrollen van een project naar een machine. Er wordt voor gezorgd dat de applicatie klaar is voor gebruik. |
| Idempotent | Iets dat meerdere keren uitgevoerd kan worden, zonder dat het resultaat veranderd. |
| Play | Bevat één of meerdere taken om uit te voeren. |
| Playbook | Bevat één of meerdere plays. |
| Provision | Het installeren van de benodigde software op een machine om er voor te zorgen dat deze gereed is voor een bepaald project. |
| Skeleton database | Een lege database met alleen de tabel structuur. |
| Vagrant | Wordt gebruikt om een virtuele ontwikkelomgeving te initialiseren. |

# 7 Bronvermelding

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

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