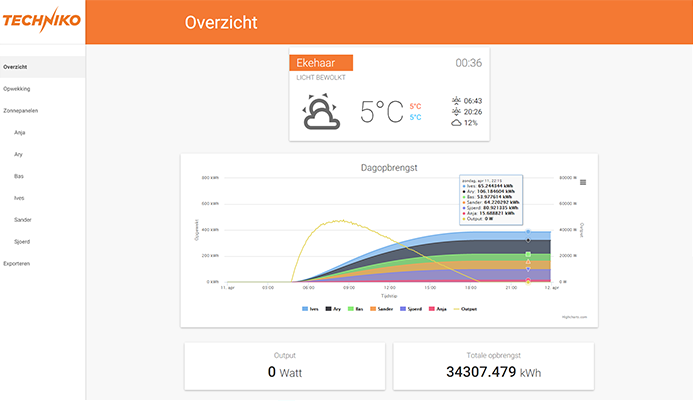
Documentatie SolarBoard

De SolarBoard is een app (bestaande uit een website, database en server) die Techniko informeert over de laatste opbrengsten van hun zonnepanelen. De website is zo geschreven dat het eenvoudig is om mobiel te bedienen. Daarnaast heeft het deze functies: een waarschuwingssysteem voor zonnepaneelgroepen die het niet doen, een mogelijkheid om vergelijkingen met vorige dagen te doen en een functie om de data te exporteren naar een .csv (af te lezen met excel).

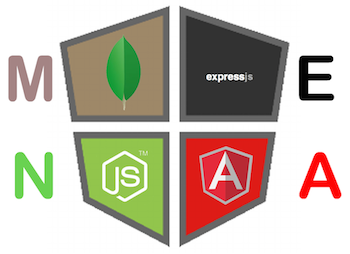


Figuur 1 Frontpage van de webapp

De website is modulair gebouwd en dus eenvoudig te bewerken door medewerkers van Techniko. Het enige wat ze nodig hebben is verstand van Javascript, HTML en CSS.

# Technische aspecten

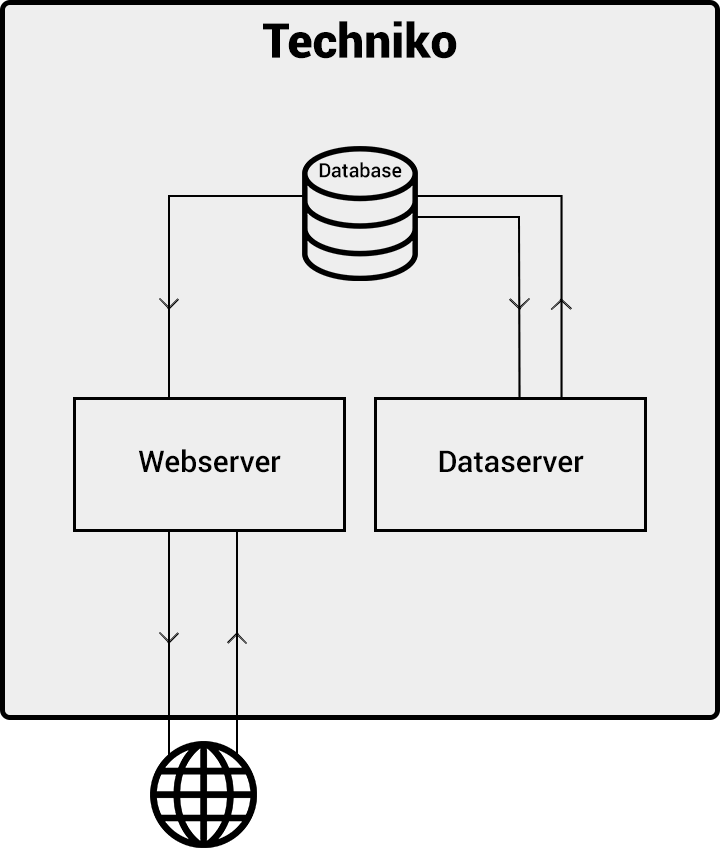
De SolarBoard is opgedeeld in twee componenten: de dataserver en de websiteserver. De dataserver is er voor het verwerken en beschikbaar maken van data, terwijl de websiteserver de verbinding voor de gebruiker regelt; het ‘host’ de website. Voor het sturen van de informatie worden arduinos gebruikt die zijn aangesloten op een Wh-meter, dit is meer toekomstbestendig dan het lezen van informatie van omvormers.   
De app zelf is geschreven met de **MEAN**-stack. Dit staat voor MongoDB, ExpressJS, AngularJS en NodeJS. Het grote voordeel van deze stack is dat de frontend én de backend in Javascript is. Hierdoor heb je geen verwarring met verschillende talen tussen beide instanties.



Figuur 2: De stack waarmee gewerkt wordt

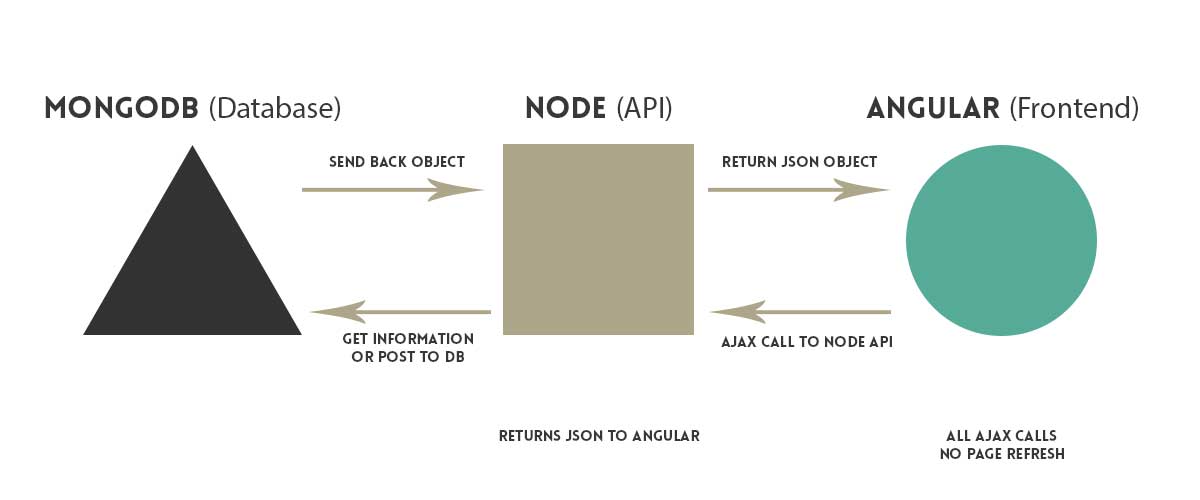
## Communicatie

Er is een duidelijk verschil tussen waar de servers mee kan communiceren. Om de informatiedatabase tegen indringers te beschermen kan de dataserver niet van buitenaf worden bestuurd, dit kan alleen gebeuren wanneer men is verbonden met het interne netwerk van Techniko. De websiteserver kan wel met instanties buiten het Technikonetwerk verbinden, er is een limitatie op de websiteserver gezet dat deze server alleen maar data van de database kan aflezen (dus niet schrijven).



Figuur 3: Overzicht van de servers in verhouding met het web en de database.

De interactie tussen de website en de database gaat (versimpeld) als volgt. Wanneer een gebruiker verbindt met de website vraagt het informatie aan, bijvoorbeeld de output van de zonnepanelen en het weer. De webserver vraag dit aan aan de database, waarna de database deze informatie terugzendt. De websiteserver geeft dit weer door aan de gebruiker en de gebruiker ‘rendert’ de webpagina ermee. Bij elkaar duurt dit vaak ongeveer 100ms, de data is namelijk erg geoptimaliseerd en geminimaliseerd. Wanneer een gebruiker een nieuwe pagina aanklikt wordt niet de hele pagina geladen, alleen maar de inhoud die de gebruiker nog niet had geladen (bijvoorbeeld het logo van Techniko). Dit is de kracht van AngularJS en de reden dat de website zo snel voelt voor de gebruiker.



Figuur 4 Beschrijving van het laden van de pagina.

De interactie tussen de dataserver en de database gaat ongeveer zoals de webserver, hier is alleen geen interactie met een gebruiker in het spel. De arduinos zenden om een bepaald tijdinterval (staat later uitgelegd onder het kopje ‘Dataverkrijging’) data naar de dataserver. Op de dataserver wordt dit omgezet naar bruikbare informatie (het omzetten van W naar Wh bijvoorbeeld) en opgeslagen in de database, waarna er nog eventueel extra data wordt opgeslagen. De dataserver kan ook de database aflezen, voor wanneer er onderhoud gepleegd moet worden. De enige functie van de dataserver is dus het verwerken en opslaan van informatie.

## Databases

De databases zijn in JSON formaat (MongoDB), dat betekent dat ze als het ware in textformaat worden opgeslagen. Het grote voordeel hiervan is dat de databases relatief minder traag worden dan andere soorten databases wanneer deze groeien. Dit is uitstekend voor Techniko, want ze hebben een database die continu in data groeit. Er zijn officieel drie databases, maar omdat drie instanties (het waarschuwingssysteem, het weersysteem en het outputsysteem) zich gedragen als een database gaan we ervan uit dat er zes databases zijn.  
De zes databases zijn:

1. Solar
2. Generated
3. GeneratedMonth
4. Weather\*
5. CurrentOutput\*
6. Alerts\*

\* Niet officieel een database

Hieronder staan de indelingen van de officiële databases.

### Solar

Bevat informatie van een zonnepaneelgroep.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Value** | **Required?** | **Default** | **Description** |
| \_id | String | TRUE |  | Unique ID |
| description | String | TRUE | "" | Description of solar group |
| peak | Number | TRUE | 0 | Highest output possible (Wp) |
| invertor | String | TRUE |  | Name of the invertor |
| location | String | TRUE |  | Location of group |
| solarpanels | Number | TRUE | 1 | Amount of solar panels |
| highestOutput | Number | FALSE | 0 | Highest output ever reached |
| totalYield | Number | FALSE | 0 | Total yield of group |
| hoursOnline | Number | FALSE | 0 | Amount of hours online since added |
| dateAdded | Date | FALSE | The date it was added | Date added to database |

### Generated

Een dagelijks bestand; het bevat info per dag over hoeveel één zonnepaneelgroep heeft gegenereerd per uur en 5 minuten. Er zitten arrays in deze database, dat betekent dat er een minidatabases inzit (die eronder ook zijn beschreven). Er zijn 24 fields van 0 tot 23, dit zijn uren. Deze uren bevatten een range van 0 tot 11, dit is dus de output van 5 minuten per vijf minuten. Voor bijvoorbeeld de output van 16:45 moet je naar field 9 en dan naar field 45/5=9. Deze manier van data opslaan is zeer efficiënt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Value** | **Required?** | **Default** | **Description** |
| \_id | String | TRUE |  | Unique ID |
| solar | String | TRUE |  | ID of solar it describes |
| date | Date | TRUE |  | Date it describes (start of day) |
| 0 to 23 (so another 23 fields after this) | Array[Number] | FALSE | [] | Hour it describes |
|  |  |  |  |  |
| **Array of numbers contains:** |  |  |  |  |
| 0 to 11 (so another 10 fields after this) | Number | FALSE | 0 | Amount of Watts generated in 5 minutes |

### GeneratedMonth

In deze database wordt een samenvatting per maand per zonnepaneelgroep opgeslagen van Generated databases. Deze database wordt door de dataserver steeds bijgewerkt wanneer er nieuwe informatie wordt opgeslagen in de Generated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Value** | **Required?** | **Default** | **Description** |
| \_id | String | TRUE |  | Unique ID |
| solar | String | TRUE |  | ID of solar it describes |
| date | Date | TRUE |  | Date it describes (start of month) |
| total | Number | FALSE | 0 | Amount generated this month (Watts) |

### CurrentOutput

Elke drie seconde wordt er een getal als output verstuurd naar de dataserver. De dataserver slaat dit niet op als bestand, maar in het geheugen op basis van naam. Wanneer de dataserver nieuwe data krijgt wordt het oude data meteen verwijderd. Deze database slaat op wanneer er een record verbroken is door een zonnepaneel in de zonnepaneelgroep database.

### Weather

Om de tien minuten haalt de dataserver het laatste weer van OpenWeatherMap.org. Deze informatie wordt in het geheugen opgeslagen en doorgegeven wanneer erom gevraagd wordt.

### Alerts

De dataserver bekijkt om het uur of elke zonnepaneel heeft gereageerd, wanneer dit het geval is verhoogt de dataserver de field ‘hoursOnline’ van de desbetreffende paneelgroep met één. Wanneer dit niet het geval is wordt er een melding gestuurd naar een emailadres en wordt er een waarschuwing opgeslagen in het geheugen. Deze waarschuwing kan door de website opgevraagd worden wanneer de pagina geladen wordt. Wanneer de zonnepaneel weer reageert wordt de melding ook weggehaald.

## API

De dataserver heeft een RESTful API om data op te slaan of door te geven aan andere instanties. De data wordt verstuurd in JSON. Paramaters van GET requests staan tussen {} en ~ betekent de host van de dataserver.

### Solar

**GET:**

|  |  |
| --- | --- |
| **URL** | **Returns** |
| ~/solars | All solar panels |
| ~/solars/{solarid} | Specific solar panel |

{solarid} = id of solar panel group

**PUT:**URL: “~/solars/{solar}”

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Value** | **Required?** | **Description** |
| id | String | TRUE | Unique ID |
| description | String | TRUE | Description of solar group |
| peak | Number | TRUE | Highest output possible (Wp) |
| invertor | String | TRUE | Name of the invertor |
| location | String | TRUE | Location of group |
| solarpanels | Number | TRUE | Amount of solar panels |

**DELETE:**URL: “~/solars/{solar}”

### Generated & GeneratedMonth

**GET:**

|  |  |
| --- | --- |
| **URL** | **Returns** |
| ~/solars/generated | All winnings information |
| ~/solars/generated/solar/{solar}/{date} | That specific day of solar panel |
| ~/solars/generated/date={date} | Every winnings info of every solar panel at that date |
| ~/solars/generated/date\_start={datestart}&date\_end={dateend}/{totalonly} | Every winnings info in the given date range. |
| ~/solars/generated/month/{date} | GeneratedMonth of given month |
| ~/solars/generated/year/{date} | All GeneratedMonths of given year |

{solar} = id of solar panel group  
{date} = date in unix epoch (seconds) UTC  
{datestart} = date in unix epoch (seconds) UTC  
{dateend} = date in unix epoch (seconds) UTC  
{totalonly} = boolean, if true only totals will be sent.

**PUT:**  
URL: “~/solars/generated/”

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Value** | **Required?** | **Description** |
| solarid | String | TRUE | ID of solar |
| generated | Number | TRUE | Amount generated in last 5 minutes |
| Date | Date | FALSE | Date of information in UNIX UTC (s), default is time on dataserver |
| invertor | String | TRUE | Name of the invertor |
| location | String | TRUE | Location of group |
| solarpanels | Number | TRUE | Amount of solar panels |

**DELETE:**URL: “~/solars/generated/{solar}/{date}” (date in Unix Epoch UTC (s))

### CurrentOutput

**GET:**

|  |  |
| --- | --- |
| **URL** | **Returns** |
| ~/solars/output | All known outputs |
| ~/solars/output/{solarid} | Get output of given solar panel |

**PUT**URL: “~/solars/output/{solarid}/{output}” – Special form of put, no parameters needed, this url will fit the bill.

### Alerts

**GET:**

|  |  |
| --- | --- |
| **URL** | **Returns** |
| ~/solars/alerts | All current alerts |

### Weather

**GET:**

|  |  |
| --- | --- |
| **URL** | **Returns** |
| ~/weather | Last loaded weather |

## Dataverkrijging

De data wordt verkregen via Wh-meters na de omvormers. Deze Wh-meters geven een stroompje wanneer er een Watt door loopt naar een arduino. Deze arduino telt hoeveel stroompjes het binnenkrijgt en stuurt om de drie seconde deze informatie naar de dataserver. Om de vijf minuten stuurt het hoeveel Watts de laatste vijf minuten erdoorheen zijn gegaan naar de dataserver (via RESTful) en reset het de counter. De arduinos hebben dus een netwerkconnectie nodig om alle data te verzenden.  
Voor de toekomstbestendigheid is het het beste om een Wh-meter te gebruiken in plaats van data aflezen van de omvormers. Dit komt omdat de omvormers met regelmaat nog van datavorm kunnen veranderen, wanneer dit gebeurt moet er weer een nieuwe decryptie ervoor geschreven worden wat weer tijd vergt. Ook wanneer Techniko nieuwe omvormers zou moeten kopen zouden ze rekening moeten houden met de omvormercode, wat niet erg flexibel is.