

abacus e*Wall* GmbH Moerser Str. 89 40667 Meerbusch

Germany

June 15th 2023

Agreement between Customer

abacus

represented by Markus Ulrich 1873 Rte de l Abadie

06730 St. Andre de la Roche France

And Developer

xxxxxx

Preamble:

abacus is looking for a skilled NodeJS full stack programmer to assist with developing the frontend for an EV charger company. The project will require integrating the frontend with AWS EC2 server-side services and then do the communication with thousands of chargers. The specific features of the frontend include user authentication, real-time data display, and payment integration (NFC-google/apple/paypal). The EV charger being used is Level 2. Ideal skills and experience for the job include proficiency in NodeJS programming, expertise in AWS EC2 services, and experience with frontend development and payment gateway integration. Further info of project scope: see addendums 1-4 that are an integral part of this agreement.

1. Subject of the Agreement

1.1 The Developer shall provide services in the form of designing and coding software for electric charging.

1.2 The tasks to be performed are described in the milestone plan and addendums 1-4).

2. Costs and Payment Conditions

2.1 The cost of services under this agreement is outlined in the Milestone Plan (Addendum 1).

2.2 The Customer shall pay the agreed amount for each milestone according to the Milestone Plan (Addendum 1) to account holder:

2.3 Upon delivering the services of a milestone to the Customer and receiving confirmation from the Customer within 7 days, the Developer shall issue an invoice to the Customer. The Customer shall pay the invoice, which confirms the completion and delivery of the services in full.

2.4 Payment is due within 10 days to the Developer's account.

3. Delivery and Acceptance of Work

3.1 The Developer will develop the source code in the server and database environment of the Customer (Addendum 2). At each milestone, the Developer will deliver the source code, including documentation, as well as the running program for testing by the Customer.

3.2 The Developer shall address any bugs or malfunctions found in the services within a period of six months from the completion of the services.

4. Confidentiality

4.1 Each party agrees to maintain the confidentiality of the other party's confidential information during the term of this Agreement and for two (2) years after its expiration or termination.

4.2 Each party shall use the other party's confidential information only as necessary to fulfill its obligations under this Agreement. Reasonable procedures shall be implemented to prevent the disclosure, unauthorized duplication, misuse, or removal of confidential information. Confidential information shall not be disclosed to employees, authorized contractors, or consultants without a need to know. Each party shall protect the other party's confidential information with the same degree of care as it exercises for its own confidential information.

4.3 The terms and conditions of this Agreement shall be treated as confidential, and neither party shall disclose them to any third party without the prior written consent of the other party. However, disclosure may be made if required by law, to legal counsel, accountants, professional advisors, banks, investors, financing sources, or in connection with the enforcement of this

Agreement or an actual or prospective merger or acquisition. In the event of required disclosure, the disclosing party shall notify the other party and make reasonable efforts to preserve the confidentiality of the Agreement.

4.4 Upon expiration of the agreement, the Developer shall delete all provided information and developed codes for the Customer from their equipment that is not part of abacus eWall infrastructure. If there are no additional or new tasks assigned by the Customer, the Developer shall confirm to the Customer that the respective data has been deleted.

5. Additional Conditions

5.1 All programming and information material developed under this agreement ("Products") shall be transferred to the Customer in its entirety and shall not be used by the Developer outside the scope of this Agreement. Ownership and copyrights of the Products belong to the Customer, not the Developer.

5.2 The Developer warrants that it will not infringe upon any third party's intellectual property rights and will not use third-party software or components unless the Customer is entitled to use such components freely, without being subject to any third party's terms and conditions.

The Developer shall not use any third-party software or components that are subject to terms of use unless approved in writing by the Customer.

5.3 Software, programming materials, documentation, and other services shall be delivered to the Customer using electronic tools of data communication such as the Internet and email. The place of delivery for the Products and other services shall be a computer located at the Customer's premises or a server with designated access defined by the Customer.

6. Duration of the Agreement

6.1 The duration of this agreement shall be as follows:

 Starting date: Upon signature of this Agreement by both parties.

 Expiration date: August 31, 2024.

6.2 Upon successful delivery of work by the Developer to the satisfaction of the Customer, both parties will discuss further engagement for project work and coding tasks, potentially leading to a long-term assignment.

7. Severability Clause

In the event that any provision of this Agreement is deemed invalid or unenforceable, a valid and enforceable provision that closely corresponds to the intended economic purpose of the invalid or unenforceable provision shall be deemed agreed upon. The validity of the remaining provisions shall remain unaffected.

8. Jurisdiction

To the extent legally permissible, the exclusive place of jurisdiction shall be the seat of abacus e*Wall* GmbH.

This Agreement is accepted by both parties on this 15th day of June 2023. Signatures

Markus Ulrich, abacus , Developer

Addendum 1

Please upload your project with the code to Github in a private repo and always push the current commits. In addition, add the user https://github.com/Niklasjkd to this repo.

Addendum 2 Milestone-Plan

Cost Duration

Milestone 1: System Design&Development 3500 EUR 3 weeks

Develop a system prototype & backend incl AWS & PI = Thunnissen-optimized and abacus eWall as demo system incl. authentication and authorization

(2.500 EUR)

Develop the Front Ends web PC/mobile friendly (800 EUR)

Write Tests for debugging each part (200 EUR)

Milestone 2: System Deployment 1.000 EUR 1 week

Deploy the system to a production environment, Conduct user acceptance testing, Train users on how to use the system, Provide ongoing technical support

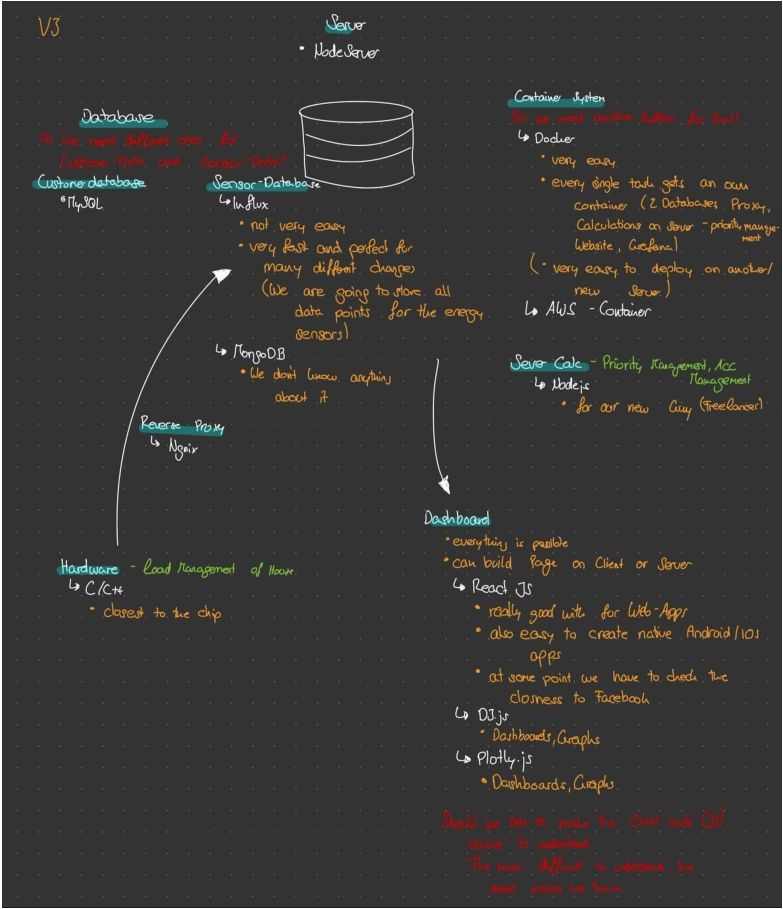
Milestone 3: NFC & Advanced Frontend Development 1. 000 EUR 3 weeks

Enable manager to integrate maps, circle lots on map and pop-up user lot line with options to send email to user and electrician, add features the garage manager wants add charts on screens to show consumption and cost per time interval for user, manager, electrician, power provider and abacus e*Wall* - Data export, especially for manager, to his billing programs

Milestone 4: Development 2.000 EUR 3 weeks

create native apps android/apple for direct Wi-Fi user mobile<> charger in garage including Billing via NFC for Apple-, Google Pay, PayPal, also scan QR code from charger display ie to link to payment site while handing over user specific data, the goal here is to simplify payment as much as possible.

All features of rules ppt and excel are minimum, but not excluding new ones coming up as the system is put into active service which means developer must solve upcoming problems, program unforeseen features ie if law requirements unknown to the customer appear



Addendum 3

abacus e*Wall* System Design:

abacus eWall infrastructure to be used and code to be stored:

Server AWS: Data Access will be given by Customer

Software being agreed on by Customer and Developer

 Database: MySQL

 Server: AWS including Container System

 Reverse Proxy: Ngingx

 Servercalculaton: NodeJS auf AWS

 Front End: ReactJs with D3.js/Plotly.js for Graphs

Any deviation of software prior to use needs to be agreed by Customer.

Addendum 4:

Garage xls and rules ppt are an integral part of this agreement, pls request per skype: markusulrich1

Addendum 5:

System description:

**IT structure abacus eWall**

Abacus eWall is a company constructing charger systems for electrical vehicles (EV) in a garage or underground garage for at least 5 cars.

1. **Components:**

**Wallbox and charging pistol**

* At the parking lot of the EV there is a wallbox with a charging pistol.
* The wallbox contains an NFC chip with a small screen (ie for QR code) to connect to the smartphone of the user.
* The wallbox has distance sensor to verify if the lot is physically available or occupied, enabling a parking guidance system that should be integrate-able into the garage-owner s system.
* The wallbox contains a WIFI module for communication to user via wifi direct and the abacus eWall server and to the PI.
* The wallbox contains an ESP32C3-chip to communicate through WIFI via garage-repeater-LAN with e-room repeater-PI.
* The wallbox communicates to the PI via websockets/MQTT via proprietary protocols, and if necessary to the power provider via ocpp which PI translates to proprietary protocol.
* The former charging module is within the wallbox now with chassis mount-20A relays.

**RCBO Cabinet**

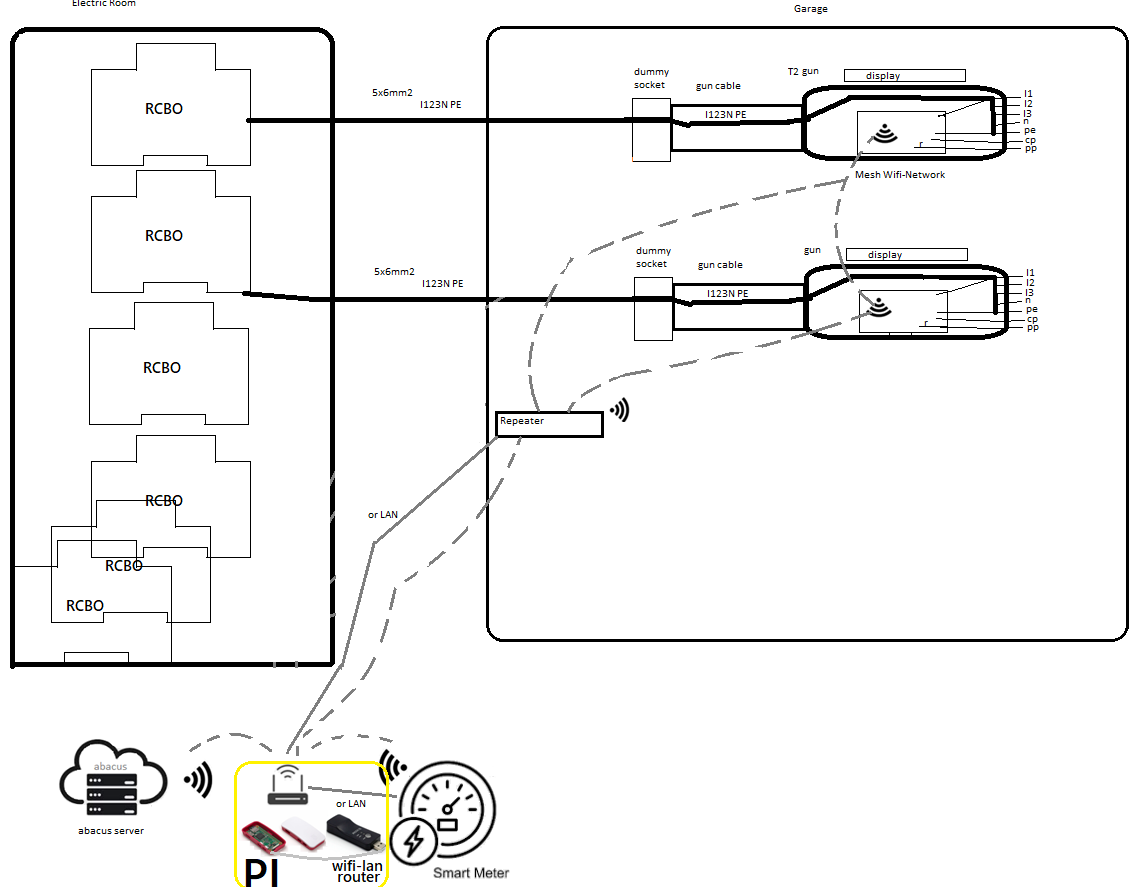
* The cabinet for RCBOs contains up to 12 RCBOs for 12 wallboxes. It can be mounted either in the room for electrical (e-Room) power where the smart meter is, or somewhere ie ceiling-mounted in the garage.
* Power entry of the cabinet is a 25-50mm2 for 200A max cable for each row (up to 12 RCBO = 6x 11kW). Power out cables are up to 12 smaller cables 5x2,5 to 6 mm2.

**Smart meter (SM)**

* In the e-room there is a smart meter measuring continuously reading the actual power consumption of the building.
* The SM is connected to a WIFI/LAN router of abacus that is connected via LAN to an IT-line provider. The WIFI signal is given to the abacus server.
* The WIFI router also connects via LAN or WIFI to the PI.

**PI**

* The PI is a small computer managing the charging of the garage for each module according to the priorities.
* The PI receives the set up of the charging system, the general information for the DLM and the setting for priorities from the abacus server in order to manage the garage indepentently of the server.
* The PI is connected to a WIFI router and communicates to all wallboxes via WIFI or LAN
* The PI connects to abacus server via WIFI
* The PI manages the DLM according to given parameters by abacus server independently from the abacus server.
* The PI, each wallbox and abacus server store all loading data. Periodically the PI exchanges information with abacus server (new lots, new priorities, lot (on/of) by manager and or electrician, charging data by lots).
* abacus server can provide statistics of the garage.
* The PI receives OCCP from 3rd parties and sends translated commands to Wallbox.



1. **abacus eWall server**

* Core of the IT is the abacus eWall (in AWS) server having all information and setting the dynamic load management (DLM) of the PI in each garage. The server can also consider power limitations by the power provider on top of limitations by the DLM. (ie get street transformer data and sets DLM according to transformer capacity.)
* The server includes the database of all garages. Each main powerline of a house is one separate identity for a garage. If there are two main powerlines providing power for one garage, the garage is separated into 2 entities for the abacus system.

**The server collects and manages the following information for each garage (identity):**

* Owner and address of building and garage
* Manager (log in), electrician of house (log in)
* Maximum capacity of building in Ampere.
* Index of capacity of regional transformer station (0 to 100 no to full use of transformer power)
* Actual used capacity of building (information by smart meter)
* Number of parking lots with fixed address (Lot 1 to …)
* Lot ‘charging’ (on/off) for administrative reasons, or for technical ones by manager
* Lot ‘charging’ (on/off) for technical reasons by house electrician
* Commentary fields for manager and electrician
* Commentary field by user for manager
* Name, mail, phone of user assigned to each Lot
* Loading power (volume) given to an EV in kW at date and time from x to z for each Lot
* Price of power in Cent per kWh at date and time
* Priority charging status defined for each Lot by user (high, normal and low priority) at cost plus, cost minus vs. regular power cost per kWh
* Charging rules / planned by user
* Sensor information of parking lot being available or occupied
* Data of charged volumes per lot of at least last 24 months

**DLM**

* The server calculates the actual capacity for EV charging (Maximum building capacity minus used capacity) for each garage at any time
* Allocation of charging power to EVs limited to maximum level agreed with power provider
* Immediate reduction of charging parallel to increased power consumption of house
* Increase of allocation in smaller steps (Puls width Modulation steps) can take up to one minute
* Charging priorities according to table (abacus rules.ppt)  
  considering:
  + Number of cars in need for charge at any time
  + Measurement of charge per phase at any time
  + Stop charging EV when car ends charging
  + Minimum charging power of a car (phase 1 to 3 or all 3)/ at one phase charge on phase 1
  + Given priorities for charging
  + Reallocate charging to groups of EVs, if there are too many cars in need  
    higher load to prio 1 Evs, lower load to others, change to other EVs every 20 minutes
  + Drive down amps/switch off L1-3 relays partially, if system recognizes building imbalance of phases > 20 Ampere

**Firmware rules:**

* Self test of each module prior to start charging
* If abacus server does not answer to actual information, or DLM does not work  
  => Stop charging when no signal from PI  
  => Stop charging when no signal from smart meter  
  => Reduce available charging power to a **defined** maximum value (0 - 10 kW (?)) at loss of connection to abacus server
* If imbalance between phases > 20 Amps: drive down amps up to shut down of relays)
* Each wallbox to save the last 1000 charging data sets
* Each SM to save 1000 data sets

**Frontend formats to manage the charging via browser:**

**Data input manager on PC/notebook:**

* Table to fill all user data per lot and to set ‘free/block’ a lot from administrative view
* To enter ie monthly installments agreed with lot user to be paid with other running costs of building
* The manager needs to approve the priority of the user in order to have a balance of high, medium and low priority users

**Data input electrician on PC/notebook/smartphone:**

* Limited view to table with lot number only to set ‘free/block’ a lot for safety reasons
* Short Comment for blocking reason

**Data input by user on smartphone via browser or App:**

* Log in register with password
* Verify Lot number, and personal information entered by manager
* Enter type of car (hybrid or EV) and model (kWh capacity)
* Ask for a defined charging priority
* Define general charging schedule
* Define charging at prices below xx Cent/ kWh, stop charging above xx Cent/ kWh
* Start adhoc charging
* Message field to manager and electrician, if something is broken or device does not work.
* NFC Connection to smartphone for direct payments *(Apple, Google Pay), for* unregistered users
* RFID card for on/off charging ie for unregistered users

**IV. Frontend output via browser (printer or interface to be defined):**

**Data output manager on PC/notebook:**

* Table with all user data and status of lot
* Sum of charged hours at average prices/ kWh to user for each period (month) and YTD
* In current year parameter of period ‘last year’ with all data of that period is needed for lot calculation (as the manager has to look into the past period for final calculation while already being in new period)
* Sum of installments and other payments by user for period and YTD
* Remaining balance / surplus of costs vs. payments
* Statistics of SM measuring (Diagram) of garage
  + Usage of power vs. maximum power in Ampere
  + Imbalance of phases (1, 2, 3) in Ampere
  + Number of shutdowns by power provider via OCCP
* User message about lot issues

**Data output electrician on PC/notebook/smartphone:**

* Lot No. , type of car and model (kWh capacity)
* Statistics of SM measuring (Diagram)
  + Usage of power vs. maximum power in Ampere (consumption vs capacity)
  + Imbalance of phases (1, 2, 3) in Ampere
* Alert of Loading due to imbalance > 20 A, electrician to manage imbalance according to power provider (reduce charging per critical phase or reduce charging on all phases)
* User message about lot issues

**Data output for user on smartphone via browser or App:**

* Log in register with password
* Information about Lot ‘available or occupied’
* When charging: Current rate of kWh charging and kW charged up to now
* When not charging: Last loading session ended at date and time with kW charged
* Table including loading sessions   
  Date and time from/to with charging at xx Cent/kWh (2 entries, if prices change at one charging cycle) at which priority  
  with totals: Number of charging sessions and kW charged at charging costs of xxx,xx €
* *Charging of not registered user with direct payment via NFC google/apple/paypal pay*

**Frontend output via printer (or interface to be defined):**

* Table of documentation for each lot in a garage/ house (if 2 or more garages in one house)
  + including all loading sessions:   
    Date and time from/to with charging at xx Cent/kWh
  + Sum of installments and other payments by user for period and YTD or closing of year
  + Remaining balance / surplus of costs vs. payments
* Statistic of number of charging sessions per lot

**Adhoc Data output manager / electrician / abacus eWall on PC/notebook/smartphone:**

* Blocking of Loading due to imbalance > 20 A , tbd: partial relay switch off (reduce charging per critical phase or reduce charging on all phases)
* Statistics of smart meter measuring (Diagram)
  + Usage of power vs. maximum power in Ampere
  + Imbalance of phases (1, 2, 3) in Ampere
* Message by user about lot that something is broken or device does not charge  
  with Statistic of that lot of last 30 days: Number of charging sessions and kW charged