Contents

[Introduction 3](#_Toc5803570)

[Technical aspects of the applications goal 3](#_Toc5803571)

[General security policy 4](#_Toc5803572)

[Sharing of sensitive data 4](#_Toc5803573)

[Passwords hashed 4](#_Toc5803574)

[Authentication stored in files 4](#_Toc5803575)

[Generating a password 4](#_Toc5803576)

[Development environment, tools and libraries 5](#_Toc5803577)

[Git repository 5](#_Toc5803579)

[Note on third party software in general 5](#_Toc5803580)

[Framework for PHP 5](#_Toc5803581)

[Development environment 5](#_Toc5803582)

[Operating system 5](#_Toc5803583)

[PHP version 5](#_Toc5803584)

[Mysql 5](#_Toc5803585)

[Composer 6](#_Toc5803586)

[NPM 6](#_Toc5803587)

[Javascript libraries 6](#_Toc5803588)

[CSS Framework 6](#_Toc5803589)

[Filesystem management 6](#_Toc5803590)

[Front-end supported browsers 7](#_Toc5803591)

[Coding style 8](#_Toc5803592)

[ORM 8](#_Toc5803593)

[Localization: languages and labels 8](#_Toc5803594)

[Label management 9](#_Toc5803595)

[Languages 9](#_Toc5803596)

[ID obfuscating 9](#_Toc5803597)

[The database 10](#_Toc5803598)

[Encoding 10](#_Toc5803599)

[Naming conventions 10](#_Toc5803600)

[Soft deletes 10](#_Toc5803601)

[Mutation dates 10](#_Toc5803602)

[Database engine 10](#_Toc5803603)

[Foreign keys 10](#_Toc5803604)

[The webservice 11](#_Toc5803605)

[Typical scenarios for the webservice 11](#_Toc5803606)

[Example front end scenarios 11](#_Toc5803607)

[Example webservice scenario 11](#_Toc5803608)

[Webservice characteristics 11](#_Toc5803609)

[Type of webservice 11](#_Toc5803610)

[Authentication 11](#_Toc5803611)

[Status codes 11](#_Toc5803612)

[The front end 12](#_Toc5803613)

[Front-end code organization 12](#_Toc5803614)

[Authentication 12](#_Toc5803615)

[Brute force protection 12](#_Toc5803616)

# Introduction

This document is the technical part of the specifications. It provides the technical context and requirements for the functional requirement document. It discusses the following parts of the project and more:

* How the development environment and tool-library look like
* Which coding style to use
* Which security measures we use
* The technical ins and outs of the different application components

## Technical aspects of the applications goal

The application will serve as a central point of existing applications within the company. The applications manages a database of client information from which other applications are dependent. It will also server as a central front-end system to manage basic administrative tasks for those applications.

The following key components of the applications can be defined:

* The database (Mysql based)
* A backend webservice (RESTful)
* Front end application (runs in browser)

The frontend will use the webservice to manage database operations. The webservice will be able to process those operations plus some requests that will come from external applications in the network.

# General security policy

## Sharing of sensitive data

Actual client or personal database data will not be published to git (like via sql dump or otherwise) or mailed or shared in any other way other that specifically agreed on. We either use the dedicated virtual machines for that or decide to use a different communication channel.

## Passwords hashed

Passwords stored in database should always be stored hashed. Plain text storage of password information is not allowed.

## Authentication stored in files

Authentication information like database or webservice credentials, will be stored in a separate env file which will be excluded from git version control. Authentication will be shared as needed via a different communication channel (like https://pwshare.com/).

## Generating a password

Passwords generated by the application should follow some basic principles: at least 10 characters from a [a-Z 0-9] alphabet. We’d like not to use special characters / punctuation characters in passwords.

For passwords generated outside of the application, please use <https://strongpasswordgenerator.com/> (tick the “avoid punctuation” box in the options panel.

# Development environment, tools and libraries

## ­­

## Git repository

Collaboration is done by using github.com. Collaborators will get an invitation to the following github repository: <https://github.com/creditdevice/cd-customer-platform.git>

A simple development environment is pre-configured to setup a development environment with PHP and Nginx. Extra required packages can be installed, please consult with us, in order to keep the provision script up to date.

## Note on third party software in general

We want to use libraries, frameworks etc. that are mature, active and popular. We want to minimize the use of many third party dependencies for very specific tasks (like many jQuery custom plugins). If unsure what to use, we have to consult. The last thing we want is conflicting or deprecated dependencies in the project.

## Framework for PHP

The PHP framework which will be used is Laravel (latest version)

<https://laravel.com/>

## Development environment

The development environment will be created by using Vagrant. De box can be setup using the vagrant file in the repository.

<https://www.vagrantup.com/>

## Operating system

The operating system to run the application is Linux (either Ubuntu or Centos). The vagrantfile from the repository already contains the necessary code to initialize the operation system when vagrant is started.

## PHP version

The PHP version that will be used is 7.2

<http://php.net/>

## Mysql

The Mysql version that will be used is Mariadb 10. We do not use specific database programming such as stored procedures, triggers and functions, nor do we use the foreign key handling. We want the application to be responsible for as much as possible.

<https://mariadb.org/>

Visual paradigm community edition is used for database modelling:

<https://www.visual-paradigm.com/download/community.jsp>

## Composer

Composer will be used to manage vendor libraries for PHP. The composer configuration for the project will be part of the git repository. The vendor libraries itself are excluded

<https://getcomposer.org/>

## NPM

Frontend javascript will be managed by npm. The configuration for the project will be part of the git repository. The vendor libraries itself are excluded.

<https://www.npmjs.com/>

## Javascript libraries

As much as possible, javascript should be build in the jQuery framework.

## CSS Framework

We want to use Bulma or Bootstrap as CSS framework. Whichever is most known to the programmer. If both are equally (un)familiar, we want to use Bulma.

The frameworks will work in conjunction with SASS

<https://bulma.io/>

<https://getbootstrap.com/>

## Filesystem management

For storing, retrieving files on server using PHP, the Flysystem abstraction library will be used.

<https://flysystem.thephpleague.com/docs/>

# Front-end supported browsers

The front-end application must run on (recent versions of) the following browsers, and give similar user experience on both Windows and Apple operating systems.

* Microsoft Edge (version 42 and up)
* Google Chrome (version 70 and up)
* Mozilla Firefox (version 60 and up)
* Apple Safari (version 12 and up, Mac version only)

There is currently no need to make a responsive version for mobile devices.

# Coding style

PSR-2 will be used as a coding style guideline and will be followed as much as possible (see <https://www.php-fig.org/psr/psr-2/> ).

Use (auto)-comments for methods, especially those which contain business logic.

# ORM

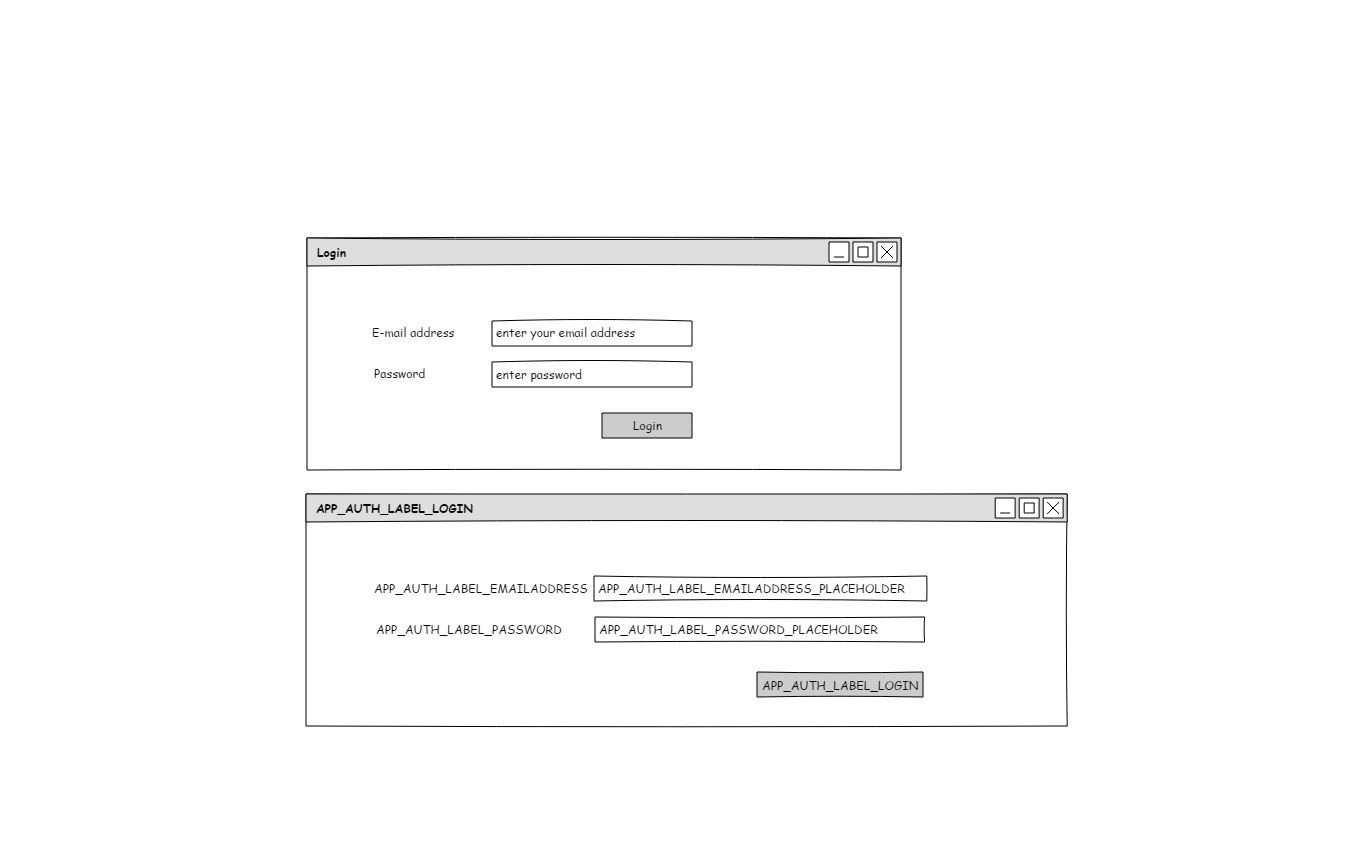
Database connection and manipulation will be done with Laravels Eloquent ORM component.

# Localization: languages and labels

The application database, front-end and webservice must be programmed in English language at all time, see coding style.

All visual messages and labels should get a system label which then will be translated by the responsible view.

See for instance mockup example of a form presented to the end user:



The first example is the translated, presentable version of a login form. The second example shows the untranslated system labels with will be programmed and translated by the view-template.

### Label management

The best practices for localization in Laravel will be followed

See <https://laravel.com/docs/5.7/localization>

### Languages

There will be two languages: English (EN) and Dutch (NL). The programmer will populate the English language file, it is the responsibility of Creditdevice to maintain the Dutch language.

# ID obfuscating

The webservice / front -end should use ID obfuscation to hide application internals, like auto-incrementing ids for tables.

One can find info and/or libraries for this

<https://hashids.org/>

And articles on Laravel for this here:

<https://jenssegers.com/64/easy-id-obfuscation-with-laravel-5>

# The database

The database is the center storage. Beside storage of blob documents (like PDF documents, attachments), all relevant data will be stored here.

The database model as proposed is found in attachment 1, the model can be found as a Mysql workbench project in the repository.

## Encoding

Database encoding is UTF-8, utf8mb4, collation utf8mb4\_unicode\_ci

## Naming conventions

We never use uppercase characters in table names or fieldnames. Phrased of multi-word table names or fieldnames will be joined together with an underscore (“\_”) character.

## Soft deletes

Some records cannot be actually deleted from the database under normal circumstances because of the dependencies that may exist. We will use soft delete fields for that in the database model (deleted\_at).

## Mutation dates

Records with mutation dates (created\_at and updated\_at) should be filled by the application upon creation (created\_at and updated\_at get the same value) and modification (updated\_at will change).

Deleted rows in general will get a deleted\_at datetime stamp, by default lists will be filtered, leaving out all entries with a deleted\_at value.

## Database engine

For all tables we use the Innodb database engine.

## Foreign keys

We do not rely on the foreign key mechanisms of Mysql, instead, the application layer will be responsible for updating and cascading of database mutations.

# The webservice

The webservice serves as the back-end layer of the application. Through the webservice, the front end is able to get,put and delete data from the database.

Furthermore, the webservice will serve as endpoint for other applications to do operations on the database.

The webservice is integrated in the same application as the front-end part of the application. They are not separated.

## Typical scenarios for the webservice

### Example front end scenarios

Front-end wants to get all the companies belonging to company X, limited to 30 rows.

Front-end wants to update a company record by adding an address

### Example webservice scenario

Application X wants to get a list of all companies for distribution channel Y, limited to 30 rows. JSON is returned.

## Webservice characteristics

### Type of webservice

The webservice application will be RESTful.

### Authentication

For authentication, JWT should be used as a means of distributing tokens. We will not be using a authentication server (like in Oauth 2.0). We will not publicly distribute tokens, the authentication should be secure but simple to implement at the same time.

### Status codes

We will use the standard status code list to inform the clients about the status of the request. The following list can be used:

<https://restfulapi.net/http-status-codes/>

# The front end

The front end is the visual application running in the browser window of the user. The user will be able to find the application by navigation to a designated URL. From here the user needs to authenticate before proceeding.

The webservice is integrated in the same application as the front-end part of the application. They are not separated.

## Front-end code organization

For the front end the Blade template component of Laravel will be used.

Minimize the use of inline javascript and css, that content will be put in separate .js and .css files which will load as needed.

## Authentication

The user will be authenticated by a authentication view displayed. The form will display fields for authentication:

* E-mail address
* Password

Since the application is targeted at a small, known, audience, a password-reset function is not required at the moment.

### Brute force protection

Upon failure to authenticate more than 5 times within 5 minutes, the IP-address should be locked out for 15 minutes duration. All amounts should be constants declared in the service which handles this so that we can finetune this afterwards.

A login attempt is always logged into the audit log tables.