Design & Implementation of Company Database for MME Subcontracting Unit

Summer Student Project Report (Year 2016; Project ID: 16476)

Author: Benedek Horváth (EN-MME-FS)

Supervisors: Alessandro Dallocchio (EN-MME-FS), Marco Garlasché (EN-MME-FS)

Abstract

The purpose of this document is to introduce the software stack designed and implemented by me, during my student project. The report includes both the project description, the requirements set against the solution, the already existing alternatives for solving the problem, and the final solution that has been implemented. Reading this document you may have a better understanding of what I was working on for eleven weeks in the summer of 2016.

1 Project description

Aim of this project is to create a new database of industrial partners, in the field of mechanical production, for daily use of the subcontracting team. The database will rely on main manufacturing technologies to establish effective research criteria.

This database is a need for the subcontracting unit, because they subcontract companies on a daily basis. Beforehand they have been keeping the contact information about these companies on contact cards. These cards have been stored in Microsoft Outlook on each unit member's account. Keeping these contact information up-to-date meant a lot of tedious work for the unit, so that they had a need for a single, shared, easy-to-use database that can be accessed from anywhere by the people who work for the subcontracting unit.

2 Requirements

Several requirements were set against the solution for this problem. First, the final product must be available on Windows PC, because that is the standard operating system used by the unit. Second, it must have full compatibility with Outlook contacts in a respect that contact cards shall be exported and imported automatically from and to Outlook without any data corruption issue. So that from the user's perspective it has to be transparent.

Third, it should have a powerful search engine that means searching the contacts not only by name, but by different fields (e.g. what capabilities the respective company has, in which country

it is, etc.) as well. Fourth, since in Outlook people can attach any file to contact cards, they would like to have this functionality in the new software as well.

Last, but not least, the software should have a nice web interface, so that it can be accessed from anywhere on any platform around the world where there is internet connection.

3 Explored available alternatives at CERN

After the requirements for the solution were discussed, several alternatives at CERN as possible solutions were explored. The first alternative was extending and customizing Outlook contact cards in a way that fits the unit's needs the best. It was soon realized that the main forms and layouts are not customizable in Outlook due to the fact that Microsoft does not provide an API for that. Besides, for keeping contacts in a synchronized place would also need to be solved.

As another option, several production information storage softwares used by different groups at CERN were explored as well. The drawbacks of these solutions were they had much more fields and capabilities than the unit would need, and they were not customizable at all, because these softwares are proprietary and developed by external companies.

So the decision was made that I will have to design and implement a full-stack solution for the problem, reusing already existing, open-source, third-party libraries and solutions wherever possible so that he would not need to reinvent the wheel by himself.

4 Solution

The architectural overview in Figure 1 shows both the main parts of the solution and also gives an overview of the interfaces these parts communicate through.

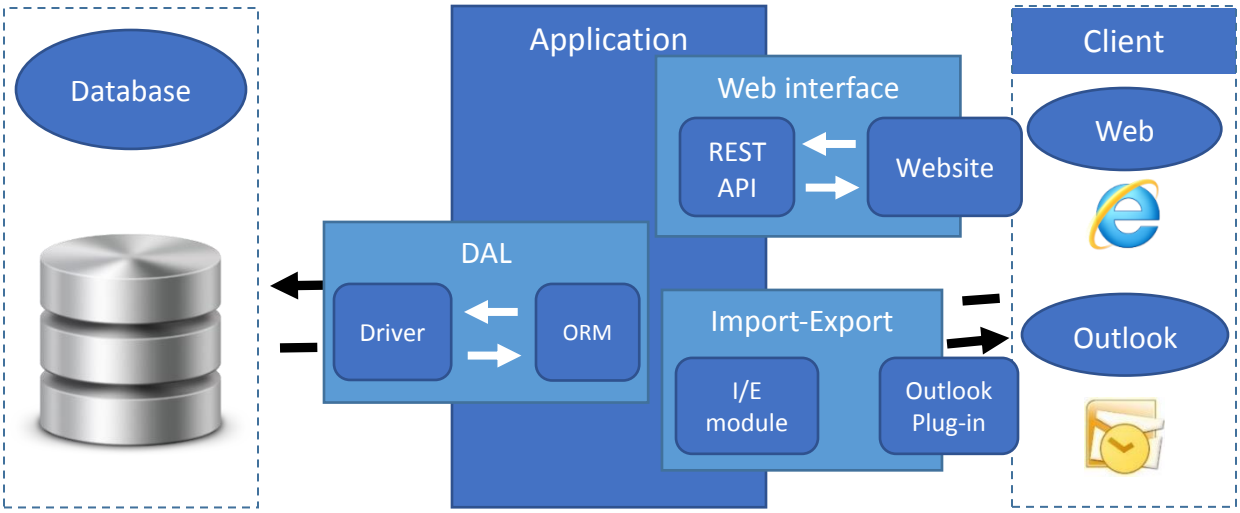

Figure 1: Architectural overview of the solution

It includes the *database backend*, which stores the information about the contact cards; the *website* which is the *frontend* to the user, where the information can be displayed and modified; the *middleware* which connects the website to the backend; and an *Outlook plug-in* which helps the user to export and import the contact cards in Outlook.

4.1 Database backend

First, the database schema had to be designed in order to get a common understanding of the relevant concepts being used in the system. The schema, depicted in Figure 2, includes all the concepts that are relevant to a contact card along with the different fields each concept has.

These concepts are translated into tables in the relational database. The tables refer to each other through foreign key (FK) attributes.

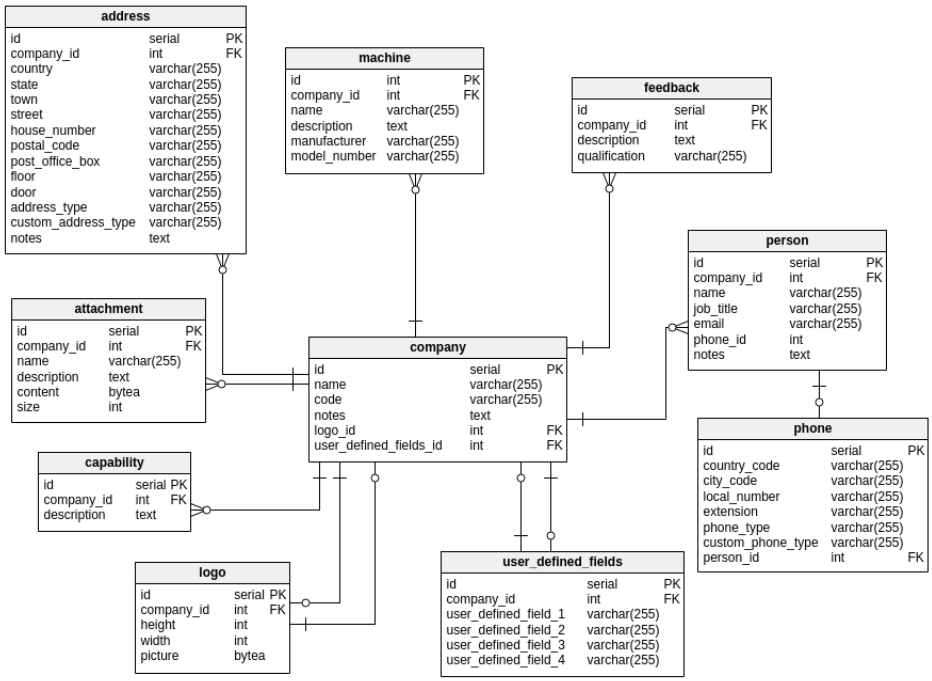

Figure 2: Database schema for the backend

For the database (DB) backend infrastructure itself, PostgreSQL database was chosen, because it was offered and maintained by CERN DB On Demand team, so that maintenance would be done by experts in this field and the subcontracting unit would not need to focus on that.

4.2 Middleware application

The purpose of the middleware application is to connect the DB with the frontend (website) so that the data stored within can be modified easily. The application was written in Java, because it is platform independent and it has lot of open-source libraries which can be reused.

The application has several main components: *objects* which resemble the data stored in the database; *manipulators* that make possible to change these objects fields for new values and to synchronize them with the DB, and *validators* which validate the fields of the objects against several rules, so that only valid data is inserted into the database.

As depicted in Figure 1 the application connects to the database through the *Data Access Layer (DAL)*. This layer makes possible for the application to be independent from the DB itself, due to the two sub-components it consists of.

The first one is the *Driver library* that is DB-dependent, which means it has to be replaced whenever the DB backend changes for a different one (e.g. from PostgreSQL to Oracle). In this case *PostgreSQL driver program* was used, because the DB backend is PostgreSQL.

The second sub-component of *DAL* is the *Object Relational Mapping (ORM)* which translates objects created by the object-oriented paradigm into SQL queries and relational data so that they can be stored in relational database. For this purpose *ORMLite* was chosen, because it is a lightweight open-source library for ORM in Java.

Besides the *DAL*, there are two other layers through that the application is accessible: the *web interface* and the *import-export* module.

The *web interface* consists of the *REST API* (REpresentational State Transfer Application Programming Interface) which enables the user to query data from the application and to send data to the application through HTTPS queries. It makes the integration easier to other applications in the future. The handler classes of the REST API queries were developed using the *Spring Framework* that is one of the most frequently used frameworks for developing enterprise Java applications.

The other main part of the *web interface* is the *website* that was developed in *Angular JS* using SPA (Single-Page Application) approach. Angular JS is one of the most widely used Java Script libraries. SPA is a design approach in which the web application loads a single HTML page and dynamically updates that page as the user interacts with the application. So that less page reloads are required which can lead to more responsive web applications.

Besides the *DAL* and the *web interface*, there is the *import-export* layer that is responsible for transferring contact cards from Outlook to the application and vice versa. It has two components.

One of them is the *import-export module* which is closely linked to the *web interface* and is responsible for processing contact card archives in the *application* and importing the cards into the *database* and vice versa. The other component is the *Microsoft Office Outlook 2013 plug-in* (add-in) that is custom extension developed for exporting and importing contact cards automatically in Outlook through a custom data structure and data format. So that transferring contact cards would be transparent from the user's perspective.

4.3 Website frontend

The website enables the user to browse the content of the contact database online, modify the contact's details (e.g. name, code, notes, addresses, machines, feedbacks, etc.), edit the respective fields, upload new attachments to companies, and to automatically export and import contacts from Outlook 2013 into the database and vice versa.

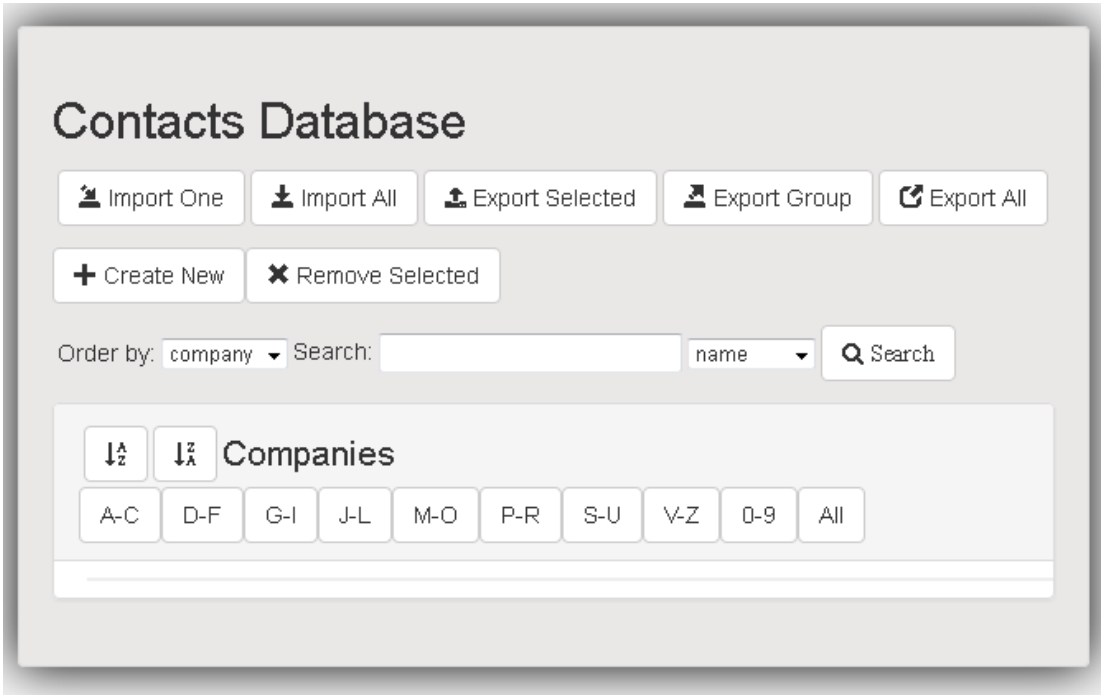

Figure 3: Main page of the website

The website has several pages. On the main page, depicted in Figure 3, the basic functionalities can be seen: import-export buttons, company creation and removal, querying companies by different fields, sorting the records by different order fields, grouping the companies by the starting letter of their names.

As soon as the user clicks on one of the letter groups in the middle, the respective companies whose names are in that range, are queried and showed in the table below. Along with the name, the country of the first address belonging to the respective company is queried as well as depicted in Figure 4.

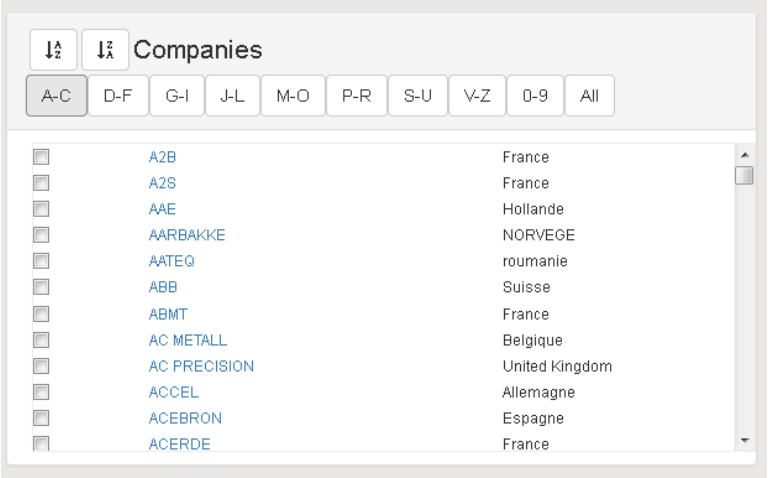

Figure 4: Partial list of the companies in range A-C

If the user clicks on a company's name in the company list, then the respective company's details are shown below the companies list. This page includes the name, code, notes and the logo (business card) of the company.

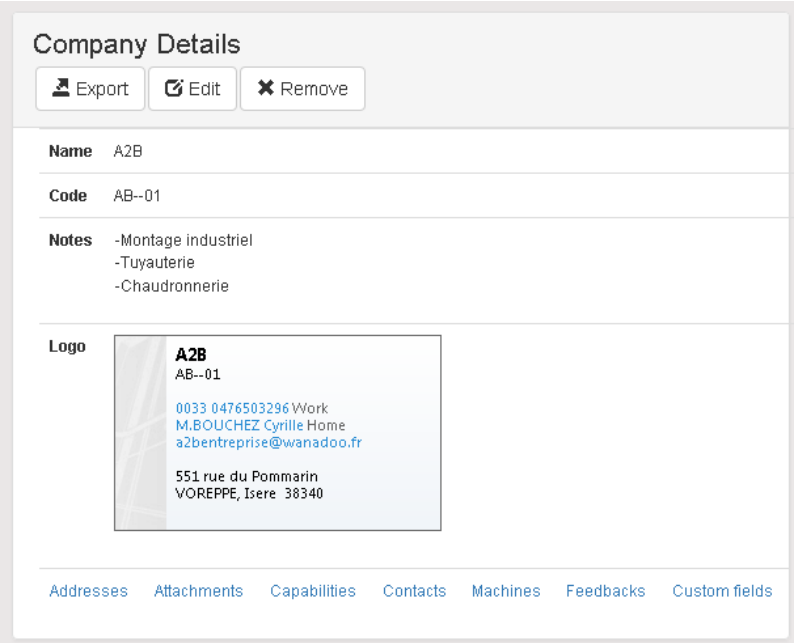

Figure 5: Company details page

At the bottom of the company details view, several other pages can be opened as well, that show the different points of interests connecting to the respective company.

E.g. when the user clicks on the 'Attachments' hyperlink at the bottom of the company details page, then the attachments, which belong to the respective company, are listed as depicted in Figure 6. An attachment can have several fields (e.g. name, description, size) and the user can download the respective file from the database.

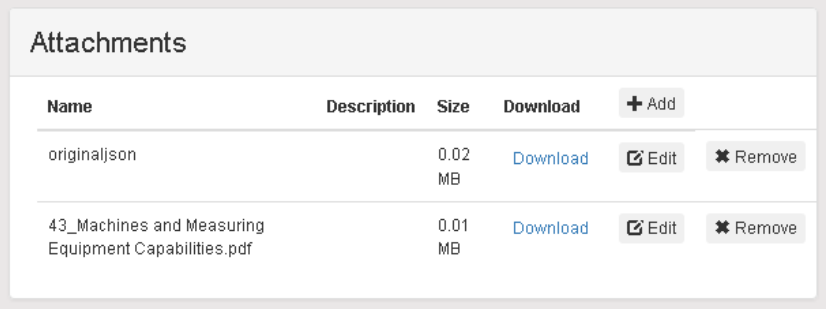

Figure 6: Attachments list

The website was deployed as a Web Service in an Apache Tomcat® container on CERN IT infrastructure which has CERN SSO and SSL enabled by default.

It means only authenticated CERN employees can access the website. Besides, all the traffic between the user's browser and the web server goes through the HTTPS protocol so that no third person can sniff the network packets. The traffic between the application and the database backend is also encrypted owing to the driver program which takes care of it. So both authentication and privacy issues were considered during design and implementation.

4.4 Microsoft Office Outlook Plug-in

A significant part of the summer student project was developing an Outlook plug-in that enables the user to automatically export and import contacts in Outlook 2013 in a custom data structure so that it will be transferable into the database through the website.

Even though Outlook has a built-in export-import functionality that was not sufficient for the project's needs. It is because the contacts can either be exported to a CSV (Comma Separated Values) file that does not contain the attachments, or to a binary Outlook data file (PST) that can have a significant size after exporting 1100 contacts. So that a custom plug-in had to be developed which would solve these issues.

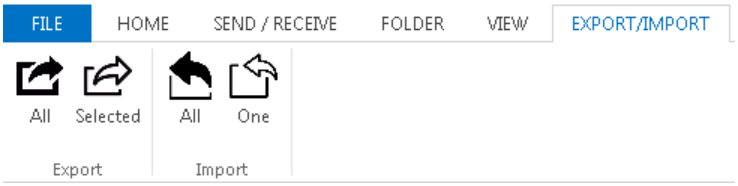

Figure 7: Export-import plug-in in Outlook

The GUI contribution of the developed plug-in is a custom ribbon on the top bar in Outlook. When the ribbon is selected and the user is on the People view in Outlook, then the features of the plug-in can be used:

- Export:
  - All: export all the contacts the user has in Outlook.
  - Selected: export only those contacts which are part of an active selection.

- Import:
    - All: import all contacts from an archive.
    - One: import only one contact that is stored in an archive.

After importing contacts from the DB into Outlook, a new file is attached to each contact card. This file contains the same information as the contact card itself, but it has several extra pieces information which are not stored on the card. E.g. additional addresses stored in the database for that contact or feedbacks given on the company or machines and capabilities the respective contact has. In this way the information stored by a card can be gradually extended.

5 Summary

During the summer student project I designed and implemented a full-stack solution for storing Outlook contact cards in a database. The solution includes both a plug-in in Outlook and an application along with an easy to use website which connects to the database and enables the user to keep contact cards information up-to-date.

Open-source third-party libraries were extensively integrated into the stack, which made the development easier and quicker. Authentication, authorization and data encryption aspects were considered during development and deployment as well.

5.1 Future Work

For the time being companies can be searched by name, capability or by country they have address in. For future work it would be a nice idea to:

- Enable searching by multiple fields at the same time.
- Search all the fields of all connecting points of interests (e.g. address, feedback, contact) for a specific keyword so that it would be more usable by the user.
- Make the search engine efficient in a mean that it takes only a few seconds to return the result of the query.

Another aspect that could be developed is predefining capabilities by labels, and letting the user to decide what these labels mean. E.g. generate from capability1 to capabilityN and decide what each capability mean (capability1 = welding, … , capability = drilling).

Besides as soon as the major version of Outlook changes, for the time being it is Outlook 2013), then the plug-in may need to be fixed accordingly if the API offered by Outlook changes as well.

5.2 Acknowledgement

I am very thankful for my supervisors, Alessandro Dallocchio and Marco Garlasché, and the Summer Student 2016 Team (HR department) for making it possible to be here. I would like to thank also for the lots of help received from the IT Help Desk while solving database or web service creation and maintenance issues.