Building Exact Online Plugins for the Salesforce Platform

Final Report

Delft, June 2014
D. Eikelenboom, T. Rensen

Preface

This report is the final report which concludes on the development of a plugin framework for the online accountancy software Exact Online, with a particular focus on implementation within Customer Relationship Management (CRM) system Salesforce.

The project was executed for the TI3800 - Bachelor End Project course within the Bachelor Computer Science and Engineering at Delft University of Technology. The project is the final project under the Bachelor phase, and was performed during the period from April 21th until the final presentation at July 4th 2014. All research and development took place at the head building of Exact International Development B.V. in Delft, The Netherlands, Molengraafsingel 33.

This report provides details on the project, requirements, design decisions, implementation, testing, process and the final product.

Especially, we would like to thank the following persons for their support during the project:

- Konrad Olczak, for coaching and steering the development process as contact person within Exact, but also for giving constructive code reviews during the project.
- Edgar Wieringa, for giving us the opportunity to perform this Bachelor End project at Exact. As well as for coordinating and steering the development process and clarifying requirements as the main stakeholder.
- Cor-Paul Bezemer, for watching the process from the TU Delft side of the project. As well as for giving extensive and constructive feedback on reports and process issues.

Dennis Eikelenboom and Tim Rensen
Delft, The Netherlands, June 2014

Summary

Exact is a global supplier of business software. Their online business solution, called Exact Online provides its users with a toolkit for Customer Relationship Management (CRM), HR, logistics and manufacturing with a particular focus on small- and medium sized businesses. Salesforce is a popular CRM system that is used by companies worldwide. Exact's recent focus with Exact Online is on creating an ecosystem of (partner) companies around Exact Online by opening up data via an Application Programming Interface (API), offering a set of mobile apps, and by offering the possibility to integrate data from Exact Online into other business applications.

For this Bachelor project, a project group consisting of two students created a plugin framework and a set of plugins that enable Exact to facilitate and develop plugins for Exact Online. The possibility to embed plugins with data from Exact Online within other business applications, will be a possible advantage for customers to choose for Exact Online above other business administration software.

In particular, Exact wanted to have a set of plugins for CRM-system Salesforce. The project team, proposed to make the assignment more abstract by using a more generic approach. An advantage of such an approach is that a platform independent set of plugins can be offered. Consequently, it will be possible to embed plugins within multiple different business applications.

The main challenge in this project was to maintain the features that an integration within a specific platform will offer, yet keeping a generic approach in order to realize a high degree of adaptability, extendability, maintainability and to allow a secure environment for data exchange. With the final prototype that the project team delivered at the end of this project, those goals have been achieved.

The final prototype of the Exact Online Plugin framework can be embedded in two different ways in another application. Firstly, it is possible to use a plugin as a 'stand-alone' version. A plugin can be embedded within another application, and show all data related to accounts in Exact Online. Secondly, it is possible to embed a plugin into another application while sending along data that belongs to the other application. These data can be for example company names or phone numbers. The plugin framework will then attempt to match the provided account data with accounts from Exact Online, and show a customized view adapted on and integrated with the 'host' application.

Matching is an essential part of the integration of external accounts with accounts from Exact Online. Algorithms that can be used to achieve this matching make use of the distance property of a set of words. In the implementation of the Exact Online Plugin framework this means that organization names are compared to see the degree of similarity between them. Moreover, the algorithms of Levenshtein, an algorithm of Jaro-Winkler and a combination of both are used in specific.

The matching module is a feature that have been thoroughly tested, because of the use of a test-driven approach for development. This approach, helped to optimize the level of acceptance for the matching algorithms. Next to unit and user test, a detailed assessment has been made on the security of the application. The creation of a threat model helped to identify potential threats from a security point of view. Based on the findings, application improvements could be made.

Concluding, the Exact Online Plugin Framework provides a flexible and maintenance-extensive way to create plugins for Exact Online. By maintaining a generic approach, it is possible to embed plugins within multiple different third-party applications, yet showing application-specific data because of a matching module that is built in.

Contents

1 Introduction

Exact[3] is a leading global supplier of business software. Exact is headquartered in Delft, the Netherlands and has been listed on the NYSE Euronext Amsterdam since June 1999. It serves their customers with industry-specific business software products on topics like manufacturing, accountancy, marketing, HR and logistics and is active in 125 countries worldwide.

Exact Online[4] is the main cloud solution and product division of Exact that is active in the field of online-based accountancy software, with 145,000 customers worldwide. By focusing on small- and medium businesses it provides its users with a toolset for Customer Relationship Management (CRM), logistics and manufacturing, centralized in one place.

Salesforce[5] is a global cloud computing company headquartered in San Francisco, California. It is best known for its Salesforce1 platform, which is a highly customizable business CRM platform. There are clients of Exact that use both Exact Online and Salesforce to support their business processes.

Exact wants to create an ecosystem of (partner) companies built around Exact Online in order to enhance the customer binding with Exact. Since several years, Exact has been focusing on opening up data from Exact Online via other ways than only via the web application. In 2011, a team has been started to work on an Application Programming Interface (API) that could be used by mobile apps to retrieve and store data from Exact Online. In 2014 a new phase has been started, in which Exact grants partner organizations access to their API. In this way, partners get eligible to create and publish apps in the Exact App Center[6], a central place where customers can find third-party applications. A combination of the App Center, collaborations with multiple partner (development) companies and the offering of apps and plugins for other platforms would contribute to the creation of an ecosystem around Exact Online.

In the remainder of this report design choices and full requirements will be clarified. In Section 2 the requirements are discussed. In Section 3 the Project Methodology will be discussed. The product implementation and architecture will be discussed in Section 4. The matching mechanism will be explained in Section 5, followed by Section 6 that will focus on testing. Section 7 concludes on the achievements made in this project according to the goals that has been set. Finalizing, Section 8 will give future work recommendations.

2 Problem Definition and Assignment

In the line with the recent App Center launch and the release of a set of mobile applications, Exact would like to have plugins available that show data from Exact Online in third-party web applications. Exact wants these plugins to be able to integrate within third-party business software that a share of their customers use next to Exact Online. The main focus within this assignment lies on the implementation of plugins within Salesforce. This section gives a description of the problem and the assignment in more detail.

2.1 Initial Assignment

The project team is required to perform a feasibility study (see Appendix B, Orientation report) to get insight in the possibilities to create plugins for Salesforce. To start with, a plugin should be created that shows outstanding items, e.g. sales invoices, from Exact Online ordered by customer accounts. When creating plugins for Salesforce does not turn out to be possible, an alternative target application should be sought for. When possible, and when the timeframe is reasonable enough, the project team should also create an additional plugin that will show revenues assigned to clients from Exact Online. However, to realize a revenue plugin, extension of the current API is required.

Exact wants to make it possible to embed a plugin into third-party applications by using only one line of Javascript code. LinkedIn[7] and Salesforce are parties that implemented a similar implementation and the project team should find out what is the best approach to achieve the same. In short, the piece of Javascript code should be able to load the complete interface of a plugin.

During the project, it should be kept in mind that Exact wants to publish the plugin for a large number of users. Therefore, architecture must be set up in a scalable way and it should not be time-extensive to reuse the implementation for another plugin. Good documentation is therefore a requisite. The subsections below give a brief description of the required plugin functionality. Descriptions are partly given by user stories, to make the requirements more clear.

2.1.1 Outstanding Items Plugin

Exact wants to show outstanding items (invoices) in third-party systems and Salesforce in specific. After implementation of this plugin, one should be able as a Salesforce user to open a relation account in Salesforce. A list and the total amount of outstanding items that are open in Exact Online should be showed upon. An outstanding item should mention the total amount, the VAT included and a due date. A more detailed requirement can be seen in the user story below.

User Story
As a SalesForce user I want to see the outstanding amounts of a given account so that I am informed about the healthieness of our relationship in my CRM system without the need to login to a different system.

1. How to demo?
   (a) Login to SF as an existing Exact Online user with rights to view financial information from that given account.
   (b) Show three invoices with their description, date, invoice number and outstanding amount including currency.
   (c) Show the total of these invoices

(d) Login as a user without rights, no info shown, no indication at all of the widget.

(e) When there are no outstanding items. Correct working of this is out of scope for now .

(f) The information is clearly identified as financial information from Exact Online. Yet its visual appearance matches the visual appearance of SF.

After implementing a prototype with all the required functionality as mentioned above, a more complete implementation has been proposed, agreed on and built.

This more complete model features the showing of all accounts that have outstanding items from Exact Online. After clicking on an account with outstanding items, a sublist will be shown showing all the items that are open from a specific relation. The plugin also features the possiblity to switch between the different 'divisions'. To be more precise, an Exact Online user may have access rights over multiple administrations, which are called divisions.

2.1.2 Revenues Plugin

Another plugin that Exact wants to create for the Salesforce platform is a plugin that shows the share of revenues in Exact Online associated to specific relation accounts. In this way, it becomes possible for a user to see the importancy of a specific customer in terms of revenue. More details on the implementation, would be provided if the project team is ready to start with this plugin. However, where the outstanding items data is already available in the Exact Online API, this is not the case for data about revenues. The project team should therefore extend the API. Since the limited time of the project, this might be out of scope and has to be decided during the project.

2.2 Assignment Revised

After the first implementation of the Outstanding Items plugin, a meeting and evaluation has been organized to assess and verify the delivered prototype. Although the requested functionality was available, a more integrated solution within Salesforce was aimed for. Basically, the prototype featured an independent plugin that did not use any Salesforce data.

Within the Salesforce application there exist customer relation cards showing information for a specific relation. It would be ideal for Exact if an integration of Exact Online Outstanding Items on these account cards could me made, so that a user can see instantly if a relation has outstanding items.

With this new goal, multiple challenges arose. As accounts from Salesforce do not necessarily have the same names as in Exact Online, a solution has to be found to match accounts from both systems. To illustrate, how do we match Salesforce account "Exact Software Development" with "Exact Software Development B.V." and how do we know if "Exact" and "Exact Software" are the same companies? The conclusion was made that this is an interesting challenge, but also a challenge that would undoubtedly require at least an additional sprint. Another challenge was to find a way to maintain the design goal from a generic implementation, yet still make a specific matching possible with Salesforce accounts.

During implementation of the matching functionality, it became obvious that extension of the API was not realistic within the time that was left. Therefore the project would include an empty plugin that gives a good example on how to extend the system with additional functionality once the API is ready. In addition, a document had to be created that guides other developers on how to realize this. This extension can be found in appendix C.

3 Project Methodology

In this section an overview is given of the time frame in which the project is performed, the software engineering methodology, required development tools and deliverables.

3.1 Time Frame and Methodology

The project team worked on the assignment for ten consecutive weeks, from the end of April untill the first week of July. Preceding the project period, there have been a number of meetings at Exact to specify the context of the project and to make decisions on methodologies used for the project.

Together with the stakeholders we agreed to work on-site at Exact for at least four days a week, during 10 weeks. Methodology to use is Scrum[8], with sprints of two weeks. On a weekly and daily basis meetings were held with stakeholders to discuss the project progress.

At the beginning of every project day a meeting was held in which the progress of each member is discussed. In these 'stand-up', or daily scrum, meetings every team member is required to mention the blocking issues that he might have, and also look forward to the planning of the specific day and the remainder of the sprint. These meetings typically take no more than 15 minutes.

As well, at the beginning of each sprint there are longer meetings with all the stakeholders in the project. These meetings have the goal to reflect on the last sprint and look forward on the planning for the next sprint(s) in order to reach the project goals and deadlines. Next to reflection, there is also time reserved for 'grooming' of tasks. Grooming is the process of estimating the time that is needed for each backlog item in a sprint. After a planning is made, subtasks for each backlog item are defined and by using Scrum Poker[9] time estimations are made for each task. These meetings take around a hour.

3.2 Environment

The project team is offered working places at Exact's headquarter in Delft for the whole project. Although, Exact offered the ability to work on their machines, we chose to keep working on our machines for mobility reasons. It was slightly unfortunate that our coach for the project was abroad in he first two weeks of our project. Therefore, all the meetings in the first sprint has to be performed virtually via videoconferencing. Setting up the Exact Online environment on our own systems required around three days, since we were unable to easily fix problems with missing dependencies. Fortunately, we received some help at this point from someone else.

At Exact, Microsoft Visual Studio 2013 is the recommended Integrated Development Environment (IDE) to work with. Exact Online is built with the Microsoft .NET framework and so will the system to implement be. The choice for an IDE like Microsoft Visual Studio is therefore logical since it integrates fluently with the language and it provides optimalized testing and code-review features. Next to Visual Studio, Exact uses a tool called Team Foundation Server (TFS) to keep track of versions in their software projects. TFS provides source code management functionality, but also features project management functionality aligned with the Scrum Methodology. During the project, TFS should be used by the project team to plan sprints on, create and assign working tasks and to keep track of source code changes. Time estimations for backlog items that are made during grooming sessions, can be easily assiged to tasks on TFS. All the mentioned features support project members in having a better overview of the development process.

3.3 Documentation

To get familiar with Exact's system architecture and to find other internal documents, the project team can make use of Synergy[10]. Synergy is Exact's internal system for sharing documents and people's contact information (HRM) within the company. It enables Exact's employees to find documentation about past projects, to find news updates, people and also it provides guidelines for coding style and documents. Documents that might be useful for the project team are those that describe the API architecture and API interface. New documents that will be delivered for this project should be uploaded to Synergy as well so that they can be found easily in the future by others.

For evaluation at TU Delft, multiple documents have to be handed in. Firstly, a document that describes the planning for the entire project must be handed in at the beginning of the project. An orientation report must describe the first research weeks, corresponding research outcome and conclusions. Finally, this final report should describe the entire project process, considerations, trade-offs, implementation and conclusions. The TU Coach was available during the projects to give feedback on each document once, so that the project team was able to incorporate feedback an hand in improved final documents. The reader can find both the Product Planning document as well as the Orientation Report as an appendix to this document.

Since the project team makes a prototype of the product, it is likely that Exact will not directly publish the result to its customers. Documentation on how to implement the delivered prototype on a live version is therefore very useful for Exact, as soon as the product becomes production-ready. A guide on how to release the project on a production server and on how to implement the plugins in a Salesforce app, is therefore required for Exact. As well a document on how to extend the system with additional plugins, would be useful for future developers that will work on the system. Both documents, supplemented with the TU Delft reports need to be uploaded to Synergy.

3.4 Presentations

At the end of the project, the final product should be presented both to the management team of Exact as well as for stakeholders at TU Delft. Ideally, the presentation at Exact will take place at the beginning of the last sprint of the project, so that feedback can be incorporated in the last sprint. The presentation at TU Delft will take place at the 4th of July.

At the moment of writing, the presentation of TU Delft is not held yet. Though, the presentation for the management team at Exact turned out to be very useful to collect feedback. Received feedback afterwards the presentation was valuable, since it manifested multiple new insights in both stakeholder wishes and possiblities from a technical point of view. The generic approach that the project team used was very appreciated, but the same approach however also created some challenges in keeping the implementation generic, while still providing specific functionality like Salesforce integration. A Javascript callback function, provided by the host application, is one of the proposed features to overcome these challenges. Afterwards, it might have been useful to have had the presentation at Exact at an earlier stage during the project. At such a late moment in the project, it is very hard to still incorporate feedback in the implementation.

For a more detailed view on the project planning, the reader is refered to the Project Planning document that can be found in appendix A.

4 Design and Implementation

In this section, an overview is given of the system architecture. During the implementation, design goals have been taken into consideration as defined at the beginning of the project. These design goals and an initial set up of the system architecture can be found in the orientation report, which is included in Appendix B. In the orientation report, the reader may also read more about research findings in detail.

4.1 Design Goals

As defined during the research part of this project (see Appendix B), multiple design goals are taken into account when implementing the Plugin Framework for Exact Online. Compared to the design goals as drawn in the research phase, a goal related to application security has been added. Overall, the design goals provided a good guidance during implementation. The design goals are listed and explained briefly below.

- *Maintainability* The application should be set up in such a way that it is easy maintainable by Exact's employees. This means that the product should be based on Microsoft frameworks and programming languages, as used by Exact for their development.
- *Scalability* Given the large amount of potential users, scalability should be ensured in the application architecture. Scalability can be concerned on hardware level, database level or on algorithm design. Also performance of external dependencies like the Exact Online API must be taken into account.
- *Reliability* Data to be shown by the application should be reliable. In addition, it has to be able to handle with errors and other exceptions, for example caused by the unavailability of applications where the system depends on.
- *Adaptability* It should be easy to extend the application with additional features or adapt existing features. A clear architecture that respects principles like the open/closed-principle and proper documentation of the system enables other people to understand how the system is composed.
- *Efficiency* Data should be handled in an efficient way. Loading lots of unnecessary data is not a good practice and should be avoided. In addition, data storage should be set up in such a way that redundancy is avoided en relations are set up in a scalable way.
- *Understandability* The system should be easy to understand for both its users as its developers. User interfaces should be kept minimal and be in line with the existing Exact user interface design. For developers, in-line comments and a clear architecture should provide a system architecture that can be understood quickly.
- *Availability* Because of the dependencies of the system on the API as well as on OAuth and other third parties like Salesforce, it must be ensured that the application can operate independently as much as possible.
- *Generic design* It is desirable that the system is set up in a generic way. This will make it possible to use the same application in multiple different contexts, e.g. the use of the same implementation for Salesforce and for SugarCRM[11].
- *Security* As the system potentially must handle with vulnerable data of Exact's customers, security is an important aspect. It should be prevented that data is breached by malicious parties. A security code review will be needed to assess the quality of code from a security perspective.

4.2 High-Level Architecture

Before a clear picture can be drawn on the high-level architecture, it is useful to emphasize the essential trade-offs that have been made before choosing a system architecture. Basically, two approaches were possible. Either creating plugins for each third-party business application or creating a system from scratch and using a more generic approach. Both approaches are discussed in the following subsection. Then a clear overview is given from the current system architecture.

4.2.1 Application-Specific or Generic Approach

As described in detail in the orientation report (see Appendix B), two possible approaches are possible to implement plugins for third-party systems – and in the first place Salesforce. On the one hand one can built a plugin based on the Salesforce App platform, on the other hand it is possible to create a system from scratch and inject the interface by means of an iframe into an empty Salesforce plugin.

Approach A - Creating a plugin based on the Salesforce App platform can be relatively fast compared to an alternative approach. Salesforce provides a development platform for developers to create apps. Apps need to written in the own 'Apex' programming language, and may access Salesforce's internal as well as external APIs. This provides flexibility for developers as organizations' own data might be used. Another advantage is that apps do automatically work on mobile devices as well. Salesforce provides secure storage for authorization keys if developers need to store authorization keys for APIs as an example.

Approach B - An alternative approach is to create a new system from scratch. This approach gives the development team the freedom to implement the system without having to cope with limitations due to the dependency on another platform. As the Salesforce1 platform makes use of a webinterface, created plugins could be integrated by injecting them into the Salesforce interface by using an iframe.

Although approach A provides a fast way to develop plugins for Salesforce applications, approach B provides more flexibility and less dependencies on other parties. Keeping in mind the design goals as listed in section 4.1, approach B is obviously the best approach to use. This approach fulfills the requirements on maintainability, security, adaptability and genericness as the plugin can be implemented from scratch by the developmen team. As every plugin is hosted at one place, the same plugin would be easily portable to other platforms. As well, maintainability only has to take place at one place. Security can be better ensured, as no data will be hosted at servers of third parties.

4.2.2 Components and Dependencies

Seen from a high level perspective, the system is built on top of the .NET framework with C# as implementation language. The Exact Online Plugin framework is a layer of shared functions that all plugins make use of. The functionalities of this layer consist of handling access tokens for API communication and handling user authorization via the authentication protocol OAuth.

In figure 1 an overview of the system is provided graphically. On itself, the system is dependent on two external sources, the Exact Online OAuth authorization system and the Exact Online API. A study on OAuth authorization and the Exact Online API can be found in the orientation report (Appendix B).

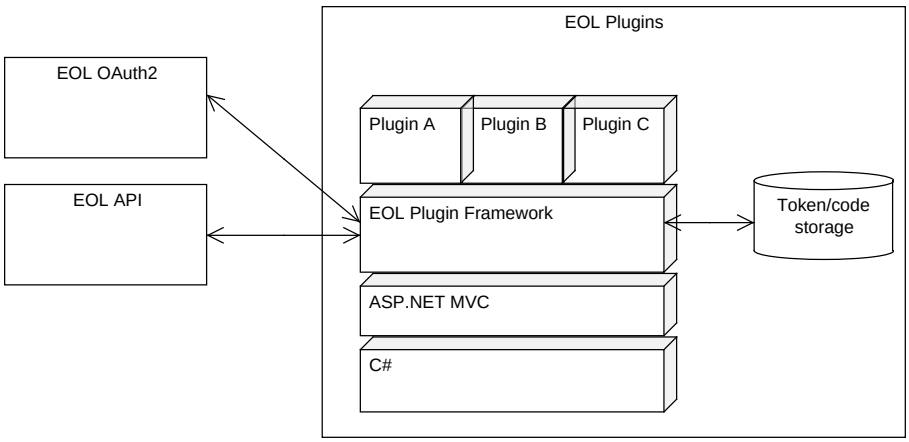


Figure 1: A high-level overview of the system.

4.3 Plugin Framework

The plugin framework layer in the system is the central part of the Exact Online Plugins system. It provides a set of shared functions that every plugin might use, and thereby makes it easy to create new plugins in relatively short time. The following subsections describe the framework's functionality in more detail.

4.3.1 Authorization

Exact Online makes use of OAuth for user authentication. An advantage of this technique (see also Appendix B) is that an user does not have to pass his credentials for Exact Online to a third-party, in this case the Exact Online Plugin framework. Instead, an access token will be passed which allows the third-party to retrieve user data from the Exact Online API.

In order to use one of the plugins, a user always has to authorize himself. From an usability perspective, it is undesirable that users have to authorize every time they want to make use of a plugin. Therefore access tokens get encrypted and are stored in a database. To identify an user, an unique user id always have to be provided within the request, when calling a plugin from an external application.

The plugin framework handles the routing of users to authorization pages, when they make use of a plugin for the first time. Figure 2 shows the Exact Online Authorization screen that is shown to plugin users.

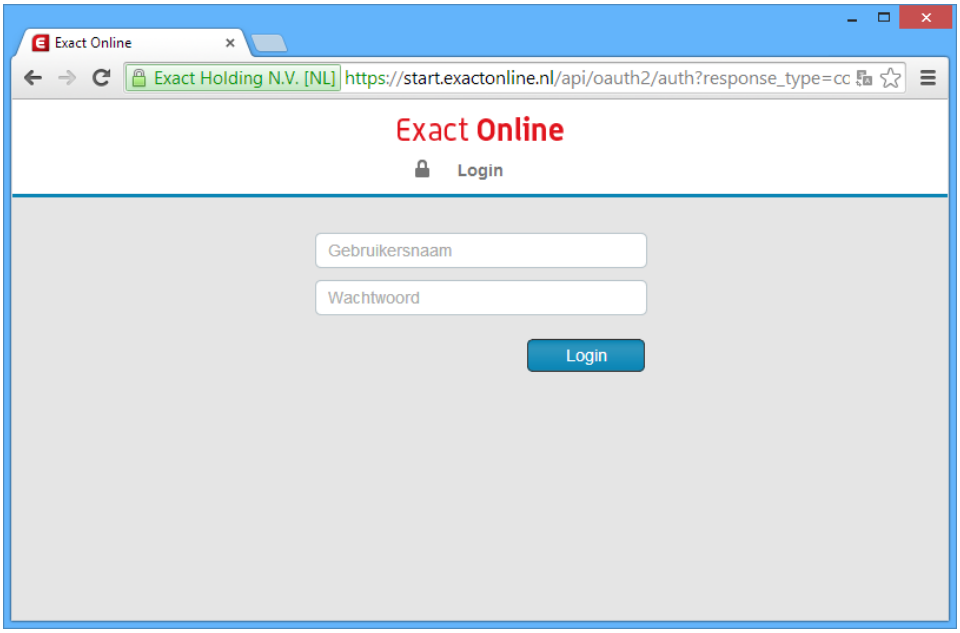


Figure 2: If a user is not authorized, the framework will redirect the user to an authorization screen of Exact Online.

4.3.2 API Communication

As every plugin has to communicate with the API in order to show data from Exact Online, it would be unnecessary complicated if every plugin needs to handle the establishment of an API connection on its own. Therefore, an API communicator is part of the shared framework functions, to set up a connection with the API and handle the refreshment of access tokens.

The functionality of the authorization and API communication part shows a large overlap, since it is required to be authorized in order to retrieve data from the API. After a user gave the plugin framework authorization to use its data, an access token is provided. Due to security concerns, this access token expires after a while, whereafter there is a need to 'refresh' a the token. The API Communicator, handles this refreshment of tokens and so provides a layer of abstraction for the actual plugins. Next to token management, the API Communicator part also takes care of a layer of abstraction in calling the api. Normally, a developer needs to get familiar with the specific uri structure of an API Call. The API Communicator however makes data retrieval easier by providing functions as 'Select' and 'Order by' that developers might use to add arguments.

4.3.3 Division Switching

An Exact Online user might have access to multiple administrations. Typically this will be multiple different organizations. A user may switch between the organizations/administrations he has access to. Administrations are also referred under the name 'division' in Exact Online.

Consequently, the same switching mechanism must be present in the plugin system as well so that users can switch between divisions as shown in Figure 3.

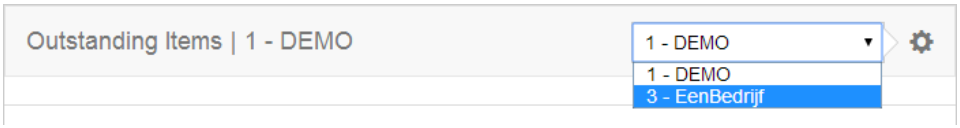


Figure 3: Division switching.

4.3.4 Account Matching

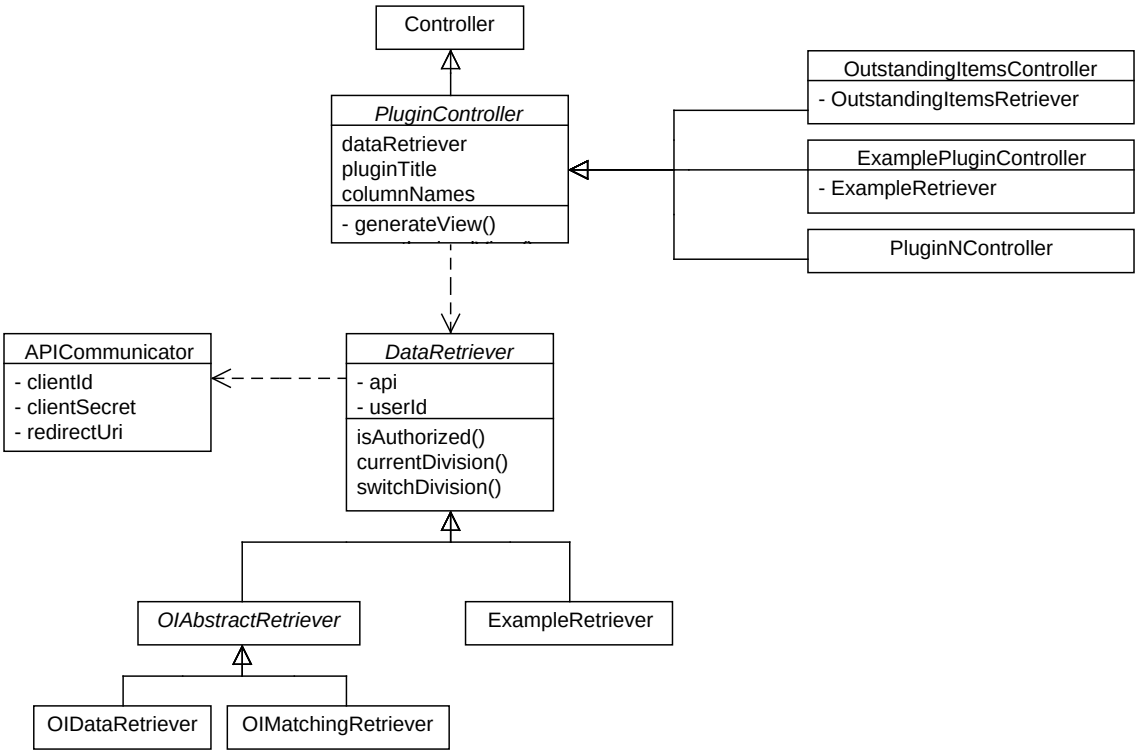
The Exact Online Plugin system is an independent system on its own, as it also can function without being implemented in a host application like Salesforce. This provides flexibility, as it is possible to integrate a plugin in multiple other applications. But it also introduces limitations, as it is hard to use data from the Salesforce application in plugins.

In order to make the data provided by plugins specific on the host application, yet maintaining a generic approach the following approach has been used. If a plugin is requested, additional data can be added to the request. This can be for example account names, organization names or phone numbers. If data is added to a request, the plugin framework will try to match the provided data with account data from Exact Online so that accounts from the host application get mapped to account from Exact Online.

Section 5 provides a detailed view on the matching module that is used to match account data in the Exact Online Plugins System.

4.4 Architecture in More Detail

The figure below gives an overview of the current system architecture. The diagram includes the plugin framework with shared functions like APICommunicator that handles the connection with the API, an abstract DataRetriever that provides communication between plugins and the API, and besides every plugin has an own pluginController that inherits from an abstract 'pluginController'. The last mentioned includes easy generation of views when someone wants to create a new plugin, but also handles routing so that unauthorized users for example will be redirected to an authorization screen. In general, for every plugin that interacts with the API a specific DataRetriever should be implemented to handle data retrieval. A new plugin can be realized by creating a new pluginController and a specific DataRetriever.



4.4.1 Building Plugins

As mentioned before, plugins should make use of an abstract DataRetriever since it takes care of the basic communication with the API. By extending the abstract retriever the basic functionality does not change, but additional API calls can be implemented which makes it possible to easily obtain new data.

If one implements a new plugin, the abstract pluginController always needs to be extended. This class will then take care of authorization and handling the refreshment of tokens. A newly created DataRetriever may be included here to extend the basic funtionality of the abstract data retriever class. This allows the developer to use additional functionality.

A more detailed version on how to extend can be found in Appendix C.

4.5 Outstanding Items Plugin

Because of the matching that needs to be done, we chose to create an abstract class with an abstract method getOutstandingItems(). This way, it was very easy to create two different retrievers: a generic one and one that takes care of the matching.

4.5.1 Stand-Alone

The stand-alone version is the intial created version of the outstanding items plugin. We chose to show a high-level list of Outstanding Items with only names and associated total amount as shown in Figure 7. By clicking on a customer row in the interface, all of the outstanding items related to that

specific customer will be shown. Items provide a description, invoice date, invoice number, currency and amount. By this way, a user can see at a glance the total amount that is still open of a specific customer. When there is need for more detailed information, the user can simply click on a name as shown in Figure 5.

Since the plugin works independently (no data from a third-party is needed), it can be easily integrated in any other application. Accounts that are shown in the interface, are the accounts as they exist in Exact Online.

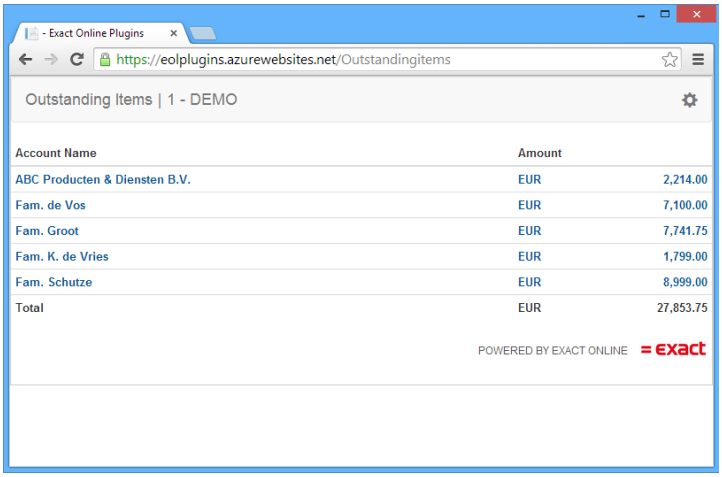


Figure 4: A basic Outstanding Items implementation.

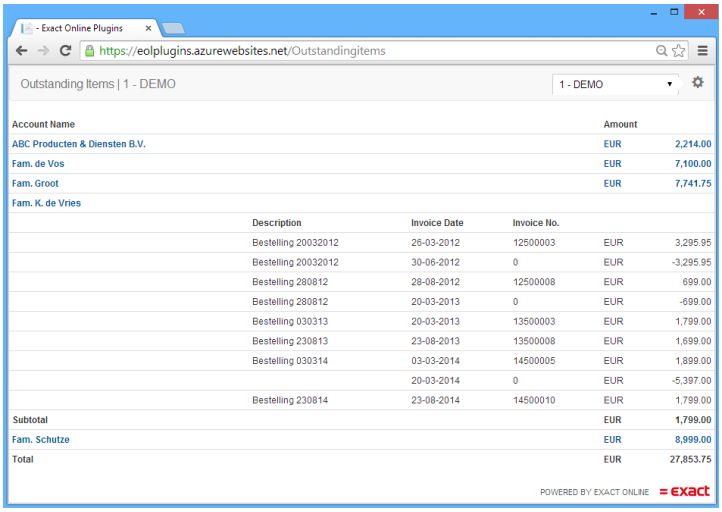


Figure 5: A basic Outstanding Items implementation with items shown for a specific account.

4.5.2 Matching External Data

To make it possible to match Exact Online account with accounts from other environments, it must be possible to send data to the plugins, which will then attempt to match these accounts. The data must be send in JSON[12] format, so that independability on other applications is avoided.

Since the matching algorithm cannot be 100 percent reliable for every match, there is a need for user verification. When the plugin is not confident if there is a match, two buttons are shown: accept and reject. The user can then decide whether to accept or reject this matching. The decision verification is stored upon in a database, so that a user only has to make the decision once. The user can revoke its decision by right clicking, on a previously made match. A popup wil then be shown to ask for a verification of the user to delete a matching.

The plugin is generic just like the stand-alone version, because of the fact that additional data must be provided in JSON format. It is possible to send data from Salesforce to this plugin, but from other environments as well.

Figure 6: Matching mechanism

During the sprint review that was following it became clear that the delivered implementation was not exactly what the stakeholders had aimed for. Though the plugin featured all the required functionality, the desire was to integrate the showing of outstanding items deeper inside the SalesForce platform. The best solution to integrate within Salesforce would have been the showing of Exact Online Outstanding items on a specific Customer relation card. A mockup of such a implementation can be seen in the picture below.

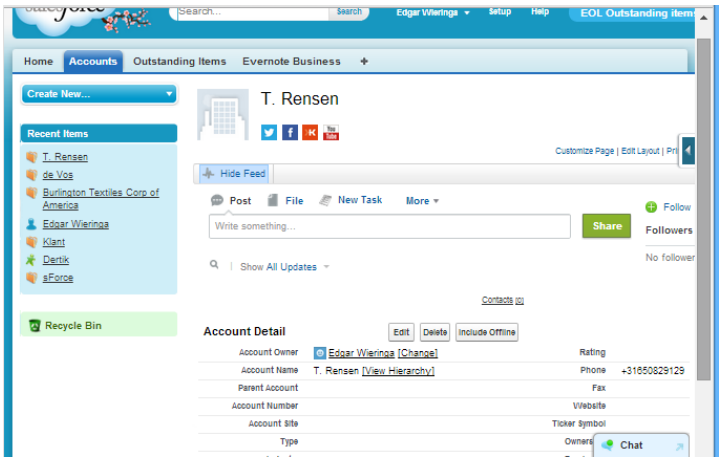


Figure 7: An account/customer relation card in Salesforce

After research through the Salesforce documentation, the project team concluded that it does not seem to be possible to plug in additional features within the account card - which is in fact a plugin on its own. The desire from Exact to fully integrate within Salesforce accounts had to be compromised for that reason. However, another way had to be proposed to integrate the outstanding items within Salesforce, as in the basic implementation accounts from both systems did not neccessarily match with eachother.

Although it was not explicitly required by Exact, we still found it a very good practice to keep a distance between the actual plugin system and the hosting business application. Therefore we had to find a

way to keep the plugins generic, including the ability to offer a specific host application dependent view. And what should not be unmentioned, we had to convince the stakeholders at Exact being this a better way of implementation than creating only a set of plugins for Salesforce.

4.6 Deployment

As a final stage before finalizing this project, the plugin framework was fully deployed on a Microsoft Azure website [1].

This suffices for testing purposes, but most likely this is a temporary solution. Publishing from Microsoft Visual Studio to an Azure website can be done easily since both products interact seamlessly with eachother. After setting up the project to interact with the Microsoft Azure servers, a developer only has to click on the Publish button, to deploy the application.

A more detailed view on deployment can be found in Appendix D.

5 Matching

To make it possible to combine accounts from different environments, some kind of matching needs to be done and it proved to be quite challenging to do this properly. Humans can make use of common sense, something that is really hard to translate to programming code.

Several algorithms are already available to calculate the similarity, or 'distance', between two strings. Firstly, the choice for the used algorithms is explained. After that, the algorithms will be described in more detail including their mathematical notation in sections 5.1, 5.2 and 5.3.

One of the best known algorithms for string distances is the Levenshtein distance (Section 5.1) which supports deleting, inserting and substituting characters.
Unfortunately, the matching results were not as good as we expected, so other algorithms needed to be taken into account.

Another algorithm is the Jaro-Winkler distance (Section 5.2) which is a variant of the Jaro distance which takes matching characters and transpositions into account and determines a score based on these aspects.
This is algorithm gave better results than Levenshtein in many cases, but still not completely what we expected.

Reverse matching (Section 5.3) made it possible to progress in the right direction. Combining these three methods, great results were achieved and no additional research was needed.

5.1 Levenshtein

A method that calculates the distance between two strings, is the Levenshtein distance. It is named after Vladimir Levenshtein who was a Russian scientist. He has done research in information theory, error-correcting codes, and combinatorial design. The algorithm was developed in 1965 and supports deleting, inserting and substituting characters.

The Levenshtein distance equals the sum of the minimum edits needed to get from one string to another string.

[1] http://azure.microsoft.com/en-us/

Example
The Levenshtein distance between 'acdfg' and 'abcef' is 3 since there is no way to do this operation with fewer than three edits:

1. acdfg $\rightarrow$ a**b**cdfg (insertion of 'b')
2. abc**d**fg $\rightarrow$ abc**e**fg (substitution of 'd' for 'e')
3. abcef**g** $\rightarrow$ abcef␣ (deletion of 'g')

Obviously, if two strings are exactly the same, no edits are needed so the distance equals 0. If one of the strings has length 0, the Levenshtein distance equals the length of the other string since that many insertions or deletions are needed. Intuitively, one could state that an edit is just a snapshot in time and is based on previous choices.

To make it possible to implement an algorithm that calculates this distance, a more formal definition is required:

The Levenshtein distance between two strings a and b is given by $lev_{a,b}(|a|, |b|)$ where

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0. \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + x \end{cases} & \text{otherwise.} \end{cases}$$

where $x = 1$ if $a_i \neq b_j$, and $x = 0$ otherwise.

$lev_{a,b}(i,j)$ can be translated as 'Levenshtein distance between strings a and b when reading the first i characters of a and the first j characters of b'.

The first part in the equation is '$\max(i,j) \quad$ if $\min(i,j) = 0$': this corresponds with the case that one of the strings has length 0, the Levenshtein equals the length of the other string as mentioned earlier.

The second part determines which choice is the most suitable. These three recursive calls represent the edits that could be made:

$lev_{a,b}(i-1,j) + 1$	deletion
$lev_{a,b}(i,j-1) + 1$	insertion
$lev_{a,b}(i-1,j-1) + x$	substitution

Example
The Levenshtein distance between two strings a and b with $|a| = 3$ and $|b| = 2$ is calculated by $lev_{a,b}(3,2)$. Since $\min(3,2) \neq 0$, the second part of the equation is used:

$$\min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + x \end{cases} = \min \begin{cases} lev_{a,b}(2,2) + 1 & \text{deletion} \\ lev_{a,b}(3,1) + 1 & \text{insertion} \\ lev_{a,b}(2,1) + x & \text{substitution} \end{cases}$$

When the Levenshtein distances of $lev_{a,b}(2,2)$, $lev_{a,b}(3,1)$ and $lev_{a,b}(2,1)$ are known, a decision is made which edit to apply.

Likewise, the 'deletion' call $lev_{a,b}(2,2)$ is based on $lev_{a,b}(1,2)$, $lev_{a,b}(2,1)$ and $lev_{a,b}(1,1)$.

Instead of using pure recursion which may result in calculating certain distances more than once, a more smart approach is taken: dynamic programming with memorization in the form of a lookup table which is filled in iteratively. Firstly the table is created with size $(|a| + 1)$ x $(|b| + 1)$. Secondly, all of the base cases are filled in, which correspond with the first row and first column of the table. Lastly, each empty cell can determine its value by looking up values at left (deletion), top (insertion) or top-left (substitution), starting off with cell $(1, 1)$. After that, the remainder of row 1 is filled in before iterating over rows $2, \ldots, |b| + 1$.

Example – Base cases
Suppose there are two strings $a = uvw$ and $b = xy$. If one compares 0 characters from string a (i.e. $i = 0$, first column) with j characters from b, the distance will equal j. Same for string b, when reading 0 characters from it (i.e. $j = 0$, first row) and i characters from a, the distance is i:

	i	0	1	2	3
j			u	v	w
0		0	1	2	3
1	x	1			
2	y	2			

Example – Substitution
Suppose $a = y$ and $b = z$. The base cases are filled in as described in previous example:

	i	0	1
j			y
0		0	1
1	z	1	

To determine the distance at position $(1, 1)$, one has to calculate $lev_{a,b}(1, 1)$. $x = 1$ since $a_1 \neq b_1$:

$$\min \begin{cases} lev_{a,b}(0,1) + 1 \\ lev_{a,b}(1,0) + 1 \\ lev_{a,b}(0,0) + 1 \end{cases} = \min \begin{cases} 2 & \text{deletion} \\ 2 & \text{insertion} \\ 1 & \text{substitution} \end{cases} = 1$$

So in this case the best edit is substituting 'y' for 'z' with cost 1:

	i	0	1
j			y
0		0	1
1	z	1	**1**

If b equals 'y' instead of 'z', then 'y' was substituted for 'y': no changes made, so

	i	0	1
j			y
0		0	1
1	y	1	**0**

Example – Deletion
Suppose that a equals 'xy' and b equals 'x'.
Base cases combined with 'x' equals 'x' (substitution):

	i	0	1	2
j			x	y
0		0	1	2
1	x	1	**0**	

This time $lev_{a,b}(2,1)$ needs to be determined:

$$\min \begin{cases} lev_{a,b}(1,1)+1 \\ lev_{a,b}(2,0)+1 \\ lev_{a,b}(1,0)+1 \end{cases} = \min \begin{cases} 1 & \text{deletion} \\ 3 & \text{insertion} \\ 2 & \text{substitution} \end{cases} = 1$$

The full lookup table becomes:

	i	0	1	2
j			x	y
0		0	1	2
1	x	1	0	**1**

Example – Insertion
Suppose that a equals 'x' and b equals 'xy'.
Base cases combined with 'x' equals 'x' (substitution):

	i	0	1
j			x
0		0	1
1	x	1	**0**
2	y	2	

$lev_{a,b}(1,2)$ equals

$$\min \begin{cases} lev_{a,b}(0,2)+1 \\ lev_{a,b}(1,1)+1 \\ lev_{a,b}(0,1)+1 \end{cases} = \min \begin{cases} 3 & \text{deletion} \\ 1 & \text{insertion} \\ 2 & \text{substitution} \end{cases} = 1$$

The full table becomes:

	i	0	1
j			x
0		0	1
1	x	1	0
2	y	2	**1**

Even if the Levenshtein distance is known, it still does not clarify the extent to which the strings are a good matching. For example, a distance of 2 on short strings means that there are not many similarities, whereas a distance of 2 on long strings is not bad at all. To solve this problem, we divided the distance by the length of the longest string which results in a score [0,1] where 0 is a perfect matching:

$$\frac{lev_{a,b}(|a|,|b|)}{\max(|a|,|b|)}$$

Example
To give an impression how the Levenshtein algorithm deals with person names, the two strings $a =$ 'John Doe' and $b =$ 'J. Doe' are matched.

The table is created and filled like explained in previous Examples. $lev_{a,b}(|a|, |b|) = lev_{a,b}(8, 6) = 3$.

		i	0	1	2	3	4	5	6	7	8
	j			J	o	h	n		D	o	e
0			**0**	1	2	3	4	5	6	7	8
1	J		1	**0**	**1**	**2**	3	4	5	6	7
2	.		2	1	1	2	**3**	4	5	6	7
3			3	2	2	2	3	**3**	4	5	6
4	D		4	3	3	3	3	4	**3**	4	5
5	o		5	4	3	4	4	4	4	**3**	4
6	e		6	5	4	4	5	5	5	4	**3**

The bold numbers represent the choices that are made to achieve the optimal distance. The three numbers in a box are the edit that needed to be done:

- Deletion of 'o' $(2, 1)$.
- Deletion of 'h' $(3, 1)$.
- Substitution of 'n' for '.' $(4, 2)$.

To get a score in [0,1]:

$$\frac{lev_{a,b}(|a|, |b|)}{\max(|a|, |b|)} = \frac{lev_{a,b}(8, 6)}{\max(8, 6)} = \frac{3}{8} = 0.375.$$

5.2 Jaro and Jaro-Winkler

An algorithm that calculates the distance between two strings is the Jaro-Winkler distance which is a variant of the Jaro distance. The Jaro distance is more suitable for string based duplication detecting, whereas the Jaro-Winkler distance is best suited and designed for short strings as person names. The Jaro-Winkler algorithm boosts the score if a certain threshold has been passed, otherwise the score remains the same.

5.2.1 Jaro Distance

The Jaro distance takes the number of matching characters and number of transpositions into account. Characters need to be in a certain range to be considered matching. The number of transpositions represent the mismatch in order *of the matching characters*.

Example
Suppose there are two strings $s =$ 'abcde' and $t =$ 'dbac'. If the matching range equals 2, 'a', 'b' and 'c' are determined as matching whereas 'd' is not since the difference in index is more than 2.

The matching characters are 'a', 'b' and 'c', but in the first string they are ordered as 'abc' and in the second string as 'bac'. The number of transpositions is 2 since the first matching character of s is

not the same as the first matching character in t ('a' $\neq$ 'b'). Same to the second matching characters ('b' $\neq$ 'a'):

s	a	b	c
t	b	a	c
	×	×	✓

Mathematically the Jaro distance d_j of strings a and b is given by:

$$d_j = \begin{cases} 0 & \text{if } m = 0. \\ \frac{1}{3}\left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m}\right) & \text{otherwise.} \end{cases}$$

where m is the number of matching characters and t is half the number of transpositions. The Jaro distance equals 1 if the two string are the same and 0 if there are no similarities at all.

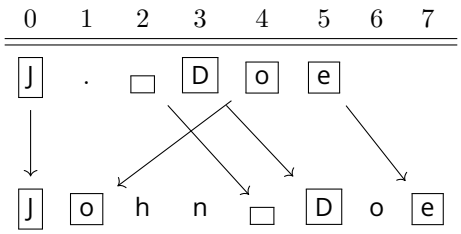
Two characters are considered matching if they are not further than

$$\left\lfloor \frac{\max(|a|, |b|)}{2} \right\rfloor - 1$$

Example
In the case of $a =$ 'J. Doe' and $b =$ 'John Doe' matching characters may not be farther than

$$\left\lfloor \frac{\max(6, 8)}{2} \right\rfloor - 1 = 3$$

'J. Doe' and 'John Doe' have five matching characters (i.e. $m = 5$) :



Note that all matching characters are not farther than 3 indices from each other.

To determine the value for t, the m matching characters are compared from s and t. In this case there are three transpositions:

s	J	␣	D	o	e
t	J	o	␣	D	e
	✓	×	×	×	✓

Thus $t = \frac{3}{2} = 1.5$.

The Jaro distance of 'J. Doe' and 'John Doe' equals

$$\frac{1}{3}\left(\frac{5}{6} + \frac{5}{8} + \frac{5 - 1.5}{5}\right) \approx 0.719$$

5.2.2 Jaro-Winkler

The Jaro-Winkler formula boosts the Jaro distance when possible and is defined as follows:

$$d_w = \begin{cases} d_j & \text{if } d_j < b_t. \\ d_j + (lp(1 - d_j)) & \text{otherwise.} \end{cases}$$

where b_t is the boost threshold which equals 0.7 in Winkler's implementation. l is the length of common prefix at the start of the string up to a maximum of 4 characters. p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. p should not exceed 0.25, otherwise the distance can become larger than 1. The standard value for this constant in Winkler's work is $p = 0.1$.

Example
The Jaro distance of 'J. Doe' and 'John Doe' equals 0.719 as explained in previous example. Since this distance succeeds the threshold 0.7, the distance will be boosted. 'J. Doe' with 'John Doe' will result in $l = 1$ because only 'J' is the same when reading from the left. The Jaro-Winkler distance will become

$$0.719 + (1 \cdot 0.1(1 - 0.719)) \approx 0.748$$

5.3 Reverse Matching

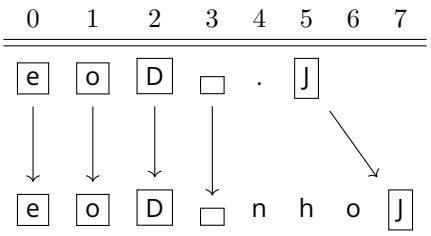
When matching 'ABCDE UVWXYZ' with 'UVWXYZ' the first six characters do not match: there is a decent chance that is this result will have a low matching score while the last name matches perfectly – with names this can happen very frequently. To solve this problem, one could take the first- and last name separately but this could give some problems with name prefixes.

The solution we came up with ourselves was reverse matching: match both 'ABCDE UVWXYZ' with 'UVWXYZ' *and* 'ZYXWVU EDCBA' with 'ZYXWVU'. By combining those outcomes a more accurate outcome is being created, since both methods have great aspects. It is worth mentioning that the Levenshtein distance remains the same when comparing reversed strings.

On the other hand, the Jaro-Winkler distance is affected when using reversed strings.

Example
The Jaro-Winkler's score of matching 'J. Doe' with 'John Doe' equals 0.748 with $m = 5$ and $t = 3$ as shown in Section 5.2.1.

When matching 'eoD .J' with 'eoD nhoJ' m equals 5 as well:



However, in this case, there are no transpositions needed, so $t = 0$. This will obviously result in a higher score:

$$\frac{1}{3}\left(\frac{5}{6} + \frac{5}{8} + \frac{5-0}{5}\right) \approx 0.819$$

With the boost of Jaro-Winkler's algorithm, the score becomes even higher: 0.892. This is mainly determined by the l which is in this case 4 (e, o, D and [space] are in the right order).

By using the avarage of 'normal' matching and reversed matching, a more realistic outcome is achieved, which was confirmed by unit testing, which will be discussed in next section.

6 Testing

In this section, an overview is given of the testing methodologies that were used during the project. The project source code has been tested with automatic tests, but not limited to only unit tests. Since the vulnerable nature of the exchanged data, security of communication is a high-listed requirement. Multiple security code reviews have been performed during the project, to reduce the risks of vulnerabilities. An initial review has been performed by Robin van Loon, one of Exact's employees who work on security. Adjustments have been made by us, and after that again verified by Exact. In particular, the following ways of testing have been used to ensure the correctness of the implemented software: automated testing, integration testing, usability testing, cross-browser testing and code reviews.

Each testing method will be discussed in the subsections below. A last subsection on code analysis describes the results of a code quality analysis performed by the Software Improvement Group (SIG).

6.1 Automated Testing

To ensure the proper working and behaviour of individual system parts like methods, the system is intensively tested by making use of automated tests, both unit tests as integration tests. Making use of unit tests, allowed us to find problems in an early stage and to simplify integration with other system parts. For some parts of the systems, like account matching, it was useful to use a test-driven approach to implement methods. By writing tests first, we were able to adapt the degree of acceptance within the algorithms to match the wanted results. In this way, we could optimize the matching module.

Some challenges emerged on the testing of system components that heavily rely on external sources like Exact's API. With unit testing it is desired to only test for the correct behaviour of single methods. When a method needs to retrieve data from an external source, you typically do not have control over what kind of data is sent. For some code components we solved these uncontrolled behaviour issues by mocking dependent classes. Still, for a large part of the API we made the decision to not test responses as the API has already intensively been tested by Exact's development teams.

6.2 Integration Testing

With integration testing we combined individual unit tests and methods in larger test methods, in order to be able to test on a higher level. This form of black-box testing was able to support us in checking the correct functionality of groups of methods. Moreover, as a result of these integration tests, an higher coverage performance could be achieved. At the most recent calculation, 70% of the system has been line-covered by tests, which is an acceptable score.

6.2.1 Testing Environments

During development, multiple environments have been used implement and test the application on. The switch of environments was due to the progress in development at the first place. As gradually more dependent components were added to the system, multiple changes of environment were needed to keep the system working correctly. An example that illustrates such a change was the embedment of a test plugin within Salesforce. Salesforce is an application that requires its users to connect with a secure connection (SSL). It turned out that it was not possible to embed our plugin without using a secure connection. Testing this environment was possible though, but only in Internet Explorer since this browser allows to access unsecure data via a secured connection. We concluded that a production server using the https protocol was an absolute requirement to deploy the application on.

Moreover, a switch of test environment was needed to test both local as live system components. During development we consecutively tested in the following environments:

- a local Internet Information Server (IIS) to run the application on;
- IIS and a local Exact Online (EOL) copy for API access;
- IIS and a local OAuth authorization component of Exact Online;
- IIS, EOL API and EOL OAuth;
- IIS, EOL API, EOL OAuth and a local database;
- IIS, EOL API/OAuth, local database and Salesforce;
- IIS, a local database and the live version of EOL within Salesforce;
- Microsoft Azure VM, an external database and the live version of EOL;

The reader might wonder why this many testing environments would have been needed during development. This was mainly due to the fact that the project team, did not have access rights to test with live components at an early stage in the development. Also, as development proceeded more components were added to the system.

At the end of the project a prototype has been delivered that runs at a Microsoft Azure virtual machine. By this way, a secure https connection can be set up between users, host applications (e.g. Salesforce) and the Exact Online plugins. A final step for Exact would be the deployment of the plugin system at an internal production server, whereafter an Salesforce app embedded with an plugin can be published in the Salesforce App Exchange.

Next to the above mentioned testing environments, the plugin interface has been tested in multiple browsers and in multiple browser versions. This Cross-browser testing turned out to be very useful for older versions of Internet Explorer, as these versions do not follow HTML and styling standards. Because of the cross-browser testing we were able to optimize our application so that it behaves correctly in all Exact Online's supported browsers.

6.3 Usability Testing

In the Scrum meetings with the stakeholders after each sprint, we showed them our progress in the development. These sessions were ideal for getting constructive advise on how to improve our application in terms of usability from a customer perspective. An important aspect was to design the plugins according to the design and the design guidelines of the Exact Online web application.

One week before the end of the project a presentation has been held for the higher management of Exact. We presented to them the design goals, implementation trade-offs and were able to give a

A1 - Injection	By using MYSQL injection it turned out to be possible to login as the last user in the database.
A2 - Broken Authentication and Session Management	For authentication it is strongly suggested to use forms authentication of ASP.NET membership ASP.NET login control or to integrate with Exact Online.
A3 - Cross-Site Scripting (XSS)	XSS was obviously present in our application. After the security review it turned out to be possible to inject code via user input to the plugin system.
A4 - Insecure Direct Object References	No threats
A5 - Security Misconfiguration	No threats at the moment. But for the live version, it would be required to set SSL configuration parameters like RequireSSL to true. Connections should be encrypted and javascript files must be minified to reduce readability.
A6 - Sensitive Data Exposure	Tokens should be encrypted to limit the impact of a data breach. Javascripts should me minified ands comments should be removed on the live version.
A7 - Missing Function Level Access Control	It was possible to unmap an map mapping of accounts of other users by changing url parameters and unauthorized call javascript functions.
A8 - Cross-Site Request Forgery (CSRF)	It seems like a possibility to (un)map accounts via CSRF attack (and edit data of other customers). Implementing [ValidateAntiForgeryToken] attributes could prevent this
A9 - Using Components with Known Vulnerabilities	No threats.
A10 - Unvalidated Redirects and Forwards	No threats.

Table 1: OWASP Security threat model

demo. After this presentation, a lot of constructive feedback could be collected from both technical as well as usability and security point of view.

6.4 Security Review

In the last sprint there was a special focus on both code quality as well as security. Together with the security division of Exact, three meetings have been organized to model potential security threats. At Exact, there are no clear guidelines regarding security, however as the security topic is becoming more and more apparent, increased attention is given on preventing possible future attacks.

Next to common code reviews, security guidelines are followed as specified by the Open Web Application Security Project (OWASP), which is a non-profit organization that has its goal set on improving the security of software. OWASP composed a top 10[13] of most common security risks to be watched for. Based on this model, a security analysis of our application was performed.

In line with the security model as described above, we could identify the following potential security threats regarding our application as shown in Table 1.

The creation of a threat model as described above made us realize how important the security topic is in current applications. Especially when you are working with sensitive data attacks should be prevented and worst-case scenarios should be assumed. In essence, this means that all sensitive data

must be encrypted at any time. In the last project-week extensive attention has been put on fixing these security issues.

6.5 Code Analysis

After each sprint, our project coach gave us feedback on the quality of code. This was not only helpful to improve the structure of the code in general, but also gave us a good direction in how to set up the application in a more scalable way. After the refactoring of 'problematic' pieces of code, additional code reviews could be requested in a handy tool in Microsoft Visual Studio. These last mentioned reviews could then be assigned to the other team member or by the project coach.

Before the start of the last sprint, the project code has been send to the Software Improvement Group[14] for a code review. In general, the project scored above average on all points. Included in the review, there were for example remarks on method length and classes with a too bright scope. Some of these points were already noticed before in the code reviews of our project coach and obviously needed to be taken in consideration. For example, the API Communicator featured both functionality that handles communication with the API as functionality for user's authorization. Recommendation following from the code reviews was consequently to split this class up in a part for authorization and a part for communication with the API.

7 Conclusions

At the beginning of the project, multiple goals were set. The majority of these goals has been reached, some even more detailed than specified and planned at the beginning of the project. In this section we will briefly discuss the goals, the final results and in particular how we view back on the process of our project.

Without taking into account limitations in time, the initial plan was to implement two plugins for Salesforce. One to show Outstanding Items in Salesforce, the other to show a list of revenues assigned to relation accounts in Salesforce. The last one could only be implemented once the Exact Online API was ready. Unfortunately there was no time left to be able to extend the API with the needed functionality. This lack of time can be ascribed to multiple factors.

Firstly, the tight planning set at the beginning of the project did not allow for multiple problems that arose during development. For example, already in the first week the set up of the complete Exact Online environment took three days longer than expected.

Secondly, during the development and the integration of the Outstanding items plugin with Salesforce it became possible that account matching was needed to achieve the required functionality. A detail that we were not aware of upfront, as we did not realize that Exact wanted to actually match accounts from Salesforce with Exact Online. Due to this new requirement, and the large amount of the time that was needed to design a appropriate implementation other features had to be compromised.

Moreover, the fact that customers desires are not known upfront is always an underestimated part in a project. Also for Exact and involved stakeholders, it was upfront only vaguely known what the final goal was for this project. This uncertainty emphasizes the importancy of a good requirements analysis and the creation of mockups of the end product. During our project, it required a lot of time to refine the implementation based on 'new' requirements, that we and stakeholders did not realize before seeing a new version of the product.

In the end, we can say that we have delivered a product that features all the functionality required at the beginning of the project, but also implemented this in such a way that it is generically available for

other applications. We showed that the plugin framework can be easily extended with other plugins (once the API has the required data available) by integrating an example plugin within the project. We were able to create an additional plugin in less than a half hour. We also ensured maintainability by setting the application up via a clear architectural modal and following Exact's code format and security guidelines. To make extendability more clear, we included an empty template plugin with the project. Concluding, we could say that we were able to deliver a tested prototype, featuring generic functionality since it integrates and matches with multiple third-party applications, and being analyzed for security threats.

8 Recommendations for Future Work

If we look back at requirements at the beginning of the project, most but not all have been fulfilled as earlier discussed in Section 7. Moreover, from a usability point of view multiple improvements can be made to enhance the current version of the system. In this section, multiple additional features are listed that can be implemented in the future to enhance the current system.

- Revenue plugin - The development of a new plugin that shows the share of revenues that a specific relation is responsible for. This feature can be implemented as soon as the Exact Online API is ready to retrieve the required data.
- Pagination - Show only a limited amount of entries from the API instead of a long list of items. From a user perspective, implementation by means of AJAX is desired as the interface then does not have to reload at every request.
- Localization - Show plugins in the language of the user. The language can be obtained after logging in via OAuth, and can be saved in the database. Translations needs to be available for every language supported by Exact Online.
- Let users manipulate data from Exact Online directly in plugins. API put functionality must be available, hence secure communication is a priority.
- Javascript callback function - Make it possible for external applications to hook on their own functionality when clicking on items in the plugin interface. An example might be the opening of a specific customer relation page in Salesforce.

References

[1] Salesforce, *Developer Documentation* Available at https://developer.salesforce.com/docs.

[2] Cihan Duruer, *Exact Online REST API - Reference documentation*. Available at https://developer.salesforce.com/docs.

[3] Exact http://www.exact.com/

[4] Exact Online http://exactonline.com/

[5] Salesforce http://www.salesforce.com/

[6] App Center https://apps.exactonline.com/nl/en-US

[7] LinkedIn https://www.linkedin.com/

[8] SCRUM method http://en.wikipedia.org/wiki/Scrum_software_development

[9] SCRUM Poker http://en.wikipedia.org/wiki/Planning_poker

[10] Synergy http://www.exact.nl/software/producten/exact-synergy-enterprise

[11] SugarCRM http://www.sugarcrm.com/

[12] JSON http://json.org/

[13] OWASP Top 10 https://www.owasp.org/index.php/Top_10_2013-Top_10

[14] Software Improvements Group http://www.sig.eu/en/

A Product Planning

Project Plan:
Exact Online Plugin framework

TI3800 Bachelorproject, course 2013-2014

Delft, June 30, 2014
D. Eikelenboom, T. Rensen

Preface

Dear stakeholder,

In this document the planning for the end project of our bachelor is described. A clear problem definition is described in this document. Moreover, this document features all the project details, stakeholders, conditions and targets that should be achieved at the end of the project. Additionally a plan is shown on the architectural design of the system. A rough planning on the complete project is also provided.

In case you have any questions regarding the project after reading this document, do not hesitate to contact us at the addresses below.

Best regards,
Dennis Eikelenboom (d.eikelenboom@student.tudelft.nl)
Tim Rensen (t.rensen@student.tudelft.nl)

Contents

1 Summary

Exact is a leading global supplier of business software. With their solutions they support over 100,000 customers, local and international companies, with the daily management of their business. Exact Online provides customers with business software in the cloud.

Recently Exact opened up parts of their system by means of an API. The goal for the project team is to make use of this API and give customers the possibility to access data from their Exact Online accounts through plugins. Since Exact is developing their products with Microsoft technology, the plugin framework should be implemented in a .NET language as well.

The project team will be supported by a project and a customer coach and will be working at working places at the Exact Office. Scrum will be used as a development method. There will be daily standup meetings with the project coach, and at the beginning of each sprint there will be a grooming session where all tasks for the coming sprint will be identified.

At the end of the project, there will be a presentation of the delivered work at the TU Delft as well as at Exact. If the work is from enough quality, is is likely that the result will be integrated and used by Exact.

2 Introduction

2.1 Emergence of this Plan

All formal information stated in this documented has been discussed and agreed on during three initial meetings at Exact preceding of the project period. Implementation and architectural details have been agreed on during the first project week together with Konrad Olczak (project coach) and Edgar Wieringa (product manager/customer).

2.2 Company Description (client)

Exact is a leading global supplier of business software. Since the beginning in 1984 the focus has shifted from supporting financial processes to developing a complete ERP offering for small and medium enterprises. Innovative solutions like Exact Globe Next, Exact Synergy Enterprise and Exact Online support over 100,000 customers local and international companies with the daily management of their business. Exact develops industry-specific on premise and SaaS solutions for manufacturing, wholesale & distribution, professional services, small business and accountancy. Exact is headquartered in Delft, the Netherlands and has been listed on the NYSE Euronext Amsterdam since June 1999.

2.3 Background and Motivation for the Project

Exact Online recently opened their data through an API to external partners. This solution creates the possibility for third parties to integrate the Exact data services into their own applications. Exact also wants to make their services available to integrate well within bigger Customer Relationship Management (CRM) systems such as Salesforce.

3 Project Assignment

3.1 Introduction

Exact wants to open up a customers' outstanding items (invoices) and a current revenue list per customer into such a system. The assignment is to set-up a framework for Exact Online plugins and develop a specific plugin for the CRM system Salesforce. The project team should explore the possibilities how to integrate such a system within the current design. An API is already available from which some functionality can be used. If functionality is needed that is not supported yet, the student should extend the API conform to the guidelines Exact set up for this. The developed solution should be accompanied with proper documentation for future development.

3.2 Contact Persons

- Edgar Wieringa (Product manager Exact)
  - edgar.wieringa@exact.com
- Konrad Olczak (Project Coach Exact)

  - konrad.olczak@exact.com
- Cor-Paul Bezemer (TU Delft Coach)
  - c.bezemer@tudelft.nl

3.3 Problem Definition

Exact Online wants to open up their system for customers outside of their main website exactonline.com. By offering multipe types of plugins, it enables their customers to integrate data from Exact into their own system.

3.4 Targets

Exact Online wants plugins that enable users to include personal data into their own software as easy as possible. The plugins and framework have to be very user friendly. It has to be easy to add plugins which offer more services other than outstanding items and revenue. A SalesForce app has to show the power of the framework.

3.5 Project Description

Exact Online is a cloud based business software platform. Exact wants to make it possible to integrate some of its functionality into another cloud solution. More specific, we think of opening-up of a customers' outstanding items and a current revenue list into such a system. The assignment is to set-up a framework for an Exact Online plug-in and develop a specific plugin for the CRM system Salesforce. The project team should explore the possibilities of integrating such a system within the current design. An API is already available from which some functionality can be used. If functionality is needed that is not supported yet, the student should extend the API conform to the guidelines Exact set up for this. Exact Online is built on top of the .NET-platform. The implementation language for the plugins depends on the other system. The developed solution should be accompanied with proper documentation for future development.

3.6 Deliverables

- Research documents
- OAuth authentication mechanism
- API interface class for plugins
- Webpage showing all available plugins
- For each plugin:
  - Contextual model
  - Plugin controller
  - Optional: customize options file
  - Needed resources (e.g. js/css)
- *Starting with:*

  - Plugin that offers outstanding items
  - Plugin that offers revenue
- Wrap 'outstanding items'-plugin into SalesForce app
- Clear documentation
- Project plan
- Final report
- Presentation at Exact and TU Delft

3.7 Conditions

Exact and the project team agreed on working onsite – and in concert it is possible to work at TU Delft or at home. At the end of this project, we have to give a presentation at Exact to show our accomplishments and to inform other stakeholders of our product. This project will be done on our own machines.

3.8 Risks

Exact uses the Microsoft .NET framework for developing their products. Since these languages are not familiar to us, it could be quite challenging to deliver proper code. Another risk is when it turns out that our goals are not possible to implement. For example, it is possible that SalesForce blocks external Javascript code, which could mean that our way to implement the Exact plugin into an app does not work.

In this case, we have to talk to the stakeholders and make comprises with what is possible. The assignment may change if such problems occur.

4 Approach

4.1 Introduction

Exact gave us the freedom to set up the project as a R&D project. This means that the initial phase of the project consists of doing research on how others implemented similar solutions, but also how Exact internally handles certain problems and how Exact set up their software architecturally. Then implementation follows and finally integration and testing.

4.2 Methods

During the project we will work with Scrum as iterative development framework. Conform to the length the sprints at Exact, we will use iterations, or sprints, of two weeks. At the beginning of each sprint there will be a grooming session, in which all the tasks for the coming week are identified. In addition, to each task points will be assigned, to represent the amount of work we expect that each task will require. Everyday, there will be a standup session together with our project coach or scrum master. In these sessions we discuss our progress, our planning for the day, and the issues that we encountered during our work.

4.3 Techniques

Exact Online is built on top the .NET framework. Because of the existing knowledge and experience among Exact's employees with Microsoft development and .NET, it is a logical choice to choose for a .NET language as main programming language. Syntactically C# is the most similar to languages that we have knowledge and experience with ourselves. Therefore we choose for this language to implement the backend of the system. Combined with ASP.NET and regular web languages as Javascript, HTML and CSS we expect ourselves to be able to produce a solid end product.

4.4 Planning

- MILESTONE I - research - 2 weeks (April 21 - May 2)
  - Project Planning (TU Delft, April 25)
  - Configure environment
  - Creating plugins for Salesforce.com
  - OAuth
  - Investige how LinkedIn achieved similar
  - OAuth Salesforce, Exact Online
  - Basic architecture
  - Orientation Report (TU Delft, April 30)
- MILESTONE II - framework - 2 weeks (May 5 - May 23)
  - Detailed architectural plan
  - Framework
  - Authentication mechanism
- MILESTONE III - outstanding items - 3 weeks (May 26 - June 13)
  - Basic app
  - Displaying outstanding items in SalesForce.com app
  - SIG Review (June 13)
- MILESTONE IV - revenue list - 3 weeks (16 June - July 4)
  - Create revenue GET API in EOL
  - Displaying revenue list in SalesForce.com app
  - Presentation Exact (June 19)
  - Final Report (TU Delft, June 27)
  - Presentation TU Delft (July 4)

5 Project design and contract

5.1 Introduction

This section points out how this project will be designed.

5.2 People Involved

In this project, there are several people involved. Edgar Wieringa is product manager at Exact and he offered us this project. He will accompany us and keeps an eye out to ensure everything goes as smoothly as possible. Konrad Olczak is an iOS app developer at Exact. He will go along with us and will help making the right decisions with the substantive aspects. The Connectivity-team of Exact plays an important role in our project since they created, among other things the API we have to extend. Cor-Paul Bezemer will be our TU Delft coach and will point out the requirements from the university (i.e. reports).

5.3 Information

Every sprint, there is a grooming session where product backlog items (PBIs) are being created. With every PBI the remaining work is estimated. The union of these PBIs define which tasks need to be completed before the ending of the corresponding sprint.

5.4 Facilities

Two workplaces are available to work with private machines. A compensation fee of EUR 364,= per person per month is offered. The employees of Exact are willing to anwser more specific questions. For example, UI Designers and Software Engineers are available to anwser ambiguities. And last but not least: free coffee is another great facility.

6 Quality Assurance

6.1 Introduction

To make sure that the delivered work is from high quality multiple methods and checks will be used and performed.

6.2 Quality

Firstly, by making use of Scrum we ensure that every two weeks there is a working, tested, version of our system. Secondly, we will write unit tests and will be using integration tests. There will be feedback on the work by the project coaches. Moreover, the project will be reviewed by the Software improvement group and the internal software validation team at Exact.

6.2.1 Documentation

Documentation will be written about the research that is done and about the implementation and architecture of the system. All code will be documented and clarified. Documents will also be published on the companies documentation and hrm system, Exact Synergy.

Multiple reports need to be handed in at the TU Delft: Project Planning, Orientation Report and Final Report which deadlines are described section 4.4.

6.2.2 Version Control

To enable the project team to use versioning, a private GitHub repository will be used to store versions of the system using git. Finally the code will also be placed on Exact's TFS server, which makes use of similar techniques.

6.2.3 Evaluation

The process and code will be evaluated during weekly and daily meetings as described above. The deliverd work and code will also be review by the Software improvement group and Exact's own internal quality team.

6.2.4 Pilots

If the delivered work is from enough quality, it is likely that Exact will integrate the Plugin framework within their current system. This will enable users of Exact Online to integrate plugins into their own internal software.

B Orientation Report

Exact Online Plugged Into SalesForce.com

Orientation Report

Delft, April 30, 2014
D. Eikelenboom, T. Rensen

Contents

1 Introduction

In this orientation report we give insight in the possible techniques that can be used to implement the desired solution: opening up customer data from business software Exact Online into Customer-relationship-management (CRM) system Salesforce.

For each design choice, we list the possible implementations and trade-offs that will be faced during this project. All decisions and implementation details for the product will have to be supported by Exact, and so introduces limitations regarding the possible alternatives.

The remainder of this document is structured as follows. In sections 2 and 3 a general introduction is given to Exact Online and Salesforce. Section 4 will describe the product with design goals and the generic framework that will be developed. A brief study on competitors and similar implementations (Exemplary solutions) shows how other parties implemented equivalent systems.

In section 5 the major dependencies will be pointed out with respect to the plugin and contains information on OAuth[1] and Application Programming Interface (API) which are needed for retrieving information from Exact Online. Section 6 is about system architecture and gives insight into the high level design of the system. Finally, our definition of done is given in section 7.

2 Exact Online

Exact is a leading global supplier of business software. Exact is headquartered in Delft, the Netherlands and has been listed on the NYSE Euronext Amsterdam since June 1999. It serves their customers with industry-specific business software products on topics like manufactoring, accountancy, marketing, HR and logistics. Exact is active in 125 countries worldwide.

Exact Online is the cloud solution that Exact offers, used by 145,000 customers worlwide focusing on small- and medium businesses. It provides its users with the tools for CRM, logistics and manufacturing, all on one central place.

To integrate more with their environment, Exact Online recently opened up their data through an API to external partners. This creates the possibility for third parties to integrate data from Exact Online into their own applications. Exact also wants to make their services available to integrate well within bigger, popular, CRM systems like Salesforce. This is what our project should aim for.

3 Salesforce

Salesforce.com is a global cloud computing company headquartered in San Francisco, California. It is best known for its Customer Relationship Management (CRM) product. Salesforce has approximated 12,000 employees and a revenue of 4 billion dollars[2].

Salesforce.com's CRM service is broken down into several broad categories: Sales Cloud Service Cloud, Data Cloud (including Jigsaw), Collaboration Cloud (including Chatter) and Custom Cloud (including Force.com), with over 100,000 customers.

Around 100,000 customers make use of Salesforce to manage their customer relation data and leads. Exact offers similar CRM features, but also offers services on other business related topics like

[1] http://oauth.net/

[2] http://en.wikipedia.org/wiki/Salesforce.com

invoicing. Logically, it may be a valuable oppertunity for Exact to integrate some of their data into Salesforce because of their existing userbase.

Salesforce has an AppExchange as well which offers multiple types of apps. In order to integrate with the Exact Online services, there is a need for the app to be able to load external data. However this step is not trivial in Salesforce because of the closed nature of the system. Normally data in Salesforce apps is stored and managed in each app's unique database tables inside Salesforce. Though, we need to make use of exernal data objects[3] in order to retrieve (and store) data from Exact Online.

4 Product

In this section, product specific information is discussed. In general, multiple design goals should be taken into account. First, these design goals are discussed, then a more product specific approach is used to clearify how these design goals will be translated into the final product.

4.1 Design Goals

Application performance is a crucial characteristic that is often a measure of how successfully the application has been constructed. Below is the list of the design goals that are aimed to achieve.

- *Maintainability* The application should be set up in such a way that it is easy maintainable by Exact's employees. This means that the product should be based on Microsoft frameworks and programming languages, as used by Exact for their development.
- *Scalability* Given the big amount of Exact Online users, and the big amount of Salesforce users, there might be a big overlap in customers that may be wanting to use our app. Therefore, scalability should be ensured in application architecture, hardware level, database design and algorithm design. The product will make use of the Exact API that is built for efficiently offering information. Data is stored in an organized way and the API allows for quick searching through data.
- *Reliability* Data to be shown by the application should be reliable. Also, it has to be able to handle with errors and other exceptions, for example caused by the unavailability of applications where the system depends on.
- *Adaptability* It should be easy to extend the application with additional features or adapt existing features. A clear architecture that respects principles like the open/closed-principle and proper documentation of the system, enables other people to understand how the system is composed.
- *Efficiency* Data should be handled in an efficient way. Loading lots of unnecessary data is not good practice and should be avoided. In addition, data storage should be set up in such a way that redundancy is avoided en relations are set up in a scalable way.
- *Understandability* The system should be easy to understand for both its users as its developers. User interfaces should be kept minimal and be in line with the existing Exact user interface design. For developers, in-line comments and a clear architecture should provide a system architecture that can be understood quickly.
- *Availability* Because of the dependencies of the system on the API as well as on OAuth and other third parties like Salesforce, it must be ensured that the application can operate independently as much as possible.

[3] http://help.salesforce.com/apex/HTViewHelpDoc?id=external_object_manage.htm

4.2 Generic Framework

The main goal of the project is to investigate the possibilities of implementing a plugin into CRM-system Salesforce. If this is possible, this system should be implemented within Salesforce. Else, an alternative target system should be chosen to integrate the plugins in.

The goal of implementing a plugin within Salesforce is possible on several ways. Salesforce provides developers with an extensive API and documentation for creating apps. While searching for possible solutions, we proposed the idea that it would be a lot more useful for Exact on the long term to have a generic framework for plugins, so that it will take less time to create another new plugin. Either contextual or for integration with another platform.

In general, two approaches are used by others to integrate plugins of an application into another (web) application. Firstly, a plugin can be written on the platform and following the conventions of the application. This means that the implementation must follow the rules and syntax of the hosting application. In the second approach, the plugin is actually hosted by the plugin creator self, and integrated (embedded) within the host application by an iframe. This latter solution is used by e.g. LinkedIn and Facebook by offering plugins to their users and so make them able to embed data on their own websites.

To achieve high independability from other systems, we convinced our stakeholders and ourselves to create a plugin that is accessible for users through an iframe and hosted on the own servers; the latter solution. The following arguments were leading for this decision: high independabiliy from host applications; low maintainance effort, no need to update multiple plugins for multiple platforms; the application can be written in any desirable language; business logic keeps inside; knowledge about the implementation language is already available among Exact's employees.

In order to set up the system as a plugin framework we came up with the idea of making use of a generic view that every plugin can use. Advantage of this set up is that when a new plugin is added, only the to be shown data has to be provided. Layout and styling is automatically applied and customizable at one place.

A similar abstraction can be created by having an abstract plugin controller from which every plugin extends. This abstract controller should handle geneneric tasks like user authorization and the creation of views. The plugin should also offer the possibility to switch between 'administrations' or 'companies' that exist in a certain Exact Online and change the view upon it.

It should be able to embed a plugin into third-party applications by implementing only one line of Javascript code. This Javascript code should load the specific plugin on that page into the third-party application.

4.3 Outstanding Items Plugin

Exact has the desire to show outstanding items (invoices) in third-party systems and Salesforce in specific.

After implementation of this plugin, one should be able as a Salesforce user to open a relation account in Salesforce. Then a list and the total amount of outstanding items that are open in Exact Online should be showed.

4.3.1 Gathering Information

The Exact Online REST API offers many resources, like "ReceivablesList", "ReceivablesListByAccount", "ReceivablesListByAgeGroup" and "ReceivablesListByAccountAndAgeGroup". It is possible to request

very specific information such as the total amount of outstanding items, but also account specific information.

However in terms of speed we consider that a request of all outstanding items in once, instead of requesting seperate items for each invoice account, is significantly faster than combining multiple requests to open items for a specific account. All individual API calls take much more time, because of the startup time (i.e. making connection). When requesting much information, there is a startup time only once and it became clear that it is much faster than seperate calls.

The API offers an option 'orderby' that enables sorting at serverside. Behind the API, information is stored in a database, which offers optimalized sorting methods. Concluding, this approach is much faster than sorting the information locally.

4.3.2 Salesforce Extension

To implement the outstanding items plugin within Salesforce, a native Salesforce app should be created. This app should embed the specific plugin by implementing the one line of Javascript code.

4.4 Revenue Plugin

Optionally, if there is enough time left, a second plugin has to be created and integrated within Salesforce. The plugin should show the current and recent revenues from an Exact Online administration.

One issue is already known when implementing this plugin, since the current API does not provide any functionality to access revenue data. The project team should extend the API's functionality before any plugin can be created.

4.5 Exemplary Solutions

Stakeholders at Exact mentioned a few implementations of plugin systems that can act as example for this project. A leading example that is proposed are the Linkedin Plugins that can be embedded with only one line of code.

LinkedIn offers multiple types of plugins[4]. One can personalize the plugin with several options like display mode (inline, icon/name, icon only, color et cetera). A small piece of default code is provided:
<script src="//platform.linkedin.com/in.js" type="text/javascript"></script>
The protocol is omitted by purpose: the environment will determine if the http or https protocol is used. For example, Chrome does not allow content over http in an https environment because of security reasons. This script loads another script dynamically, which takes care of the communication with LinkedIn's API.

Some plugins – like Member Profile – do not require authorization, so the information can be obtained anonymously. The only data that needs to be provided is the unique identifier of the requested data, for example the public URI of an user. For other plugins, like 'Full Member Profile', an API key must be provided which is simply done by setting a variable:
<script src="//platform.linkedin.com/in.js" type="text/javascript">
api_key: the_api_key_goes_here
</script>

[4] https://developer.linkedin.com/plugins

By setting variables in Javascript, some essential data can be send to the server which will take these settings into account when generating the right information in the plugin.

Exact collaborated with LinkedIn in 2010 as a launching partner for company pages. Exact Online uses the company page tool as a way to match entrepeneurs with for example accountants. It is possible to see which accountant in a client's professional network is working with Exact Online, allowing them to connect and work together within Exact Online.

An example that we found ourselves is the application that Evernote offers to their Salesforce users. Evernote is an application that let its users make notes about everything at everytime, either on their desktop or mobile phone. They created an app, available in the Salesforce app store that let their users interact with their own system. Evernote implemented a Salesforce plugin using the same techniques as we intend to. They host their plugin on their own servers, and then embed it into Salesforce using an iframe. This shows that a similar implementation must be possible.

5 Dependencies

The system rests on multiple other systems to able to functionate correctly. For each dependency, needed techniques and implementation trade-offs are discussed in this section. As earlier mentioned in the design goals, there should be aimed for as less coupling between components as possible, the same counts for these dependencies. At least, plugins should still be able to work if Salesforce is unavailable.

5.1 Exact Online

The plugin system will make use of two different existing components from Exact Online. An authorization part that implements the OAuth authentication standard, and an API that can be used in combination with OAuth to retrieve user specific data from Exact Online. f

5.1.1 REST API

There are two ways to communicate with Exact Online's API: authorization (OAuth) and authentication (Basic Authentication). Basic Authentication lets users access their own data by signing in with their normal username and password. OAuth allows access for third parties to the same resources but most of the time with limitations. A short overview of frequently used API calls are described in the appendix. The Basic Authentication will be used at first, because of testing purposes. Once the OAuth authentication parts is functional, authentication will only happen by means of OAuth given the security it provides to users' personal data.

5.1.2 OAuth

To receive an authorization code, an Exact Online user needs to log in at the authorization website of Exact Online. With this authorization request, a client_id, redirect_uri and response_type needs to provided by the developer. After logging in and granting access to the developer, the user will be redirected back to the redirect_uri with an authorization code. The authorization code is valid for 1 year. After this time, the user needs to log in again to obtain a new code. The authorization can be exchanged *once* for an access token, and with this access token a refresh token is provided as well. The developer can request a new access token with the request token.

The access token has to be passed in the header of every API request. After some time (currently 10 minutes) the access token will expire. As mentioned earlier, using the refresh token one is able to retrieve a new access token.

When the user decides to revoke access to the developer, the last-mentioned can not request new access tokens anymore from that moment.

5.1.3 OAuth Applied

The flow diagram below (figure 1) shows how the plugin framework is going to communicate with both the Exact Online authorization server and the API in order to get access tokens and the requested user data.

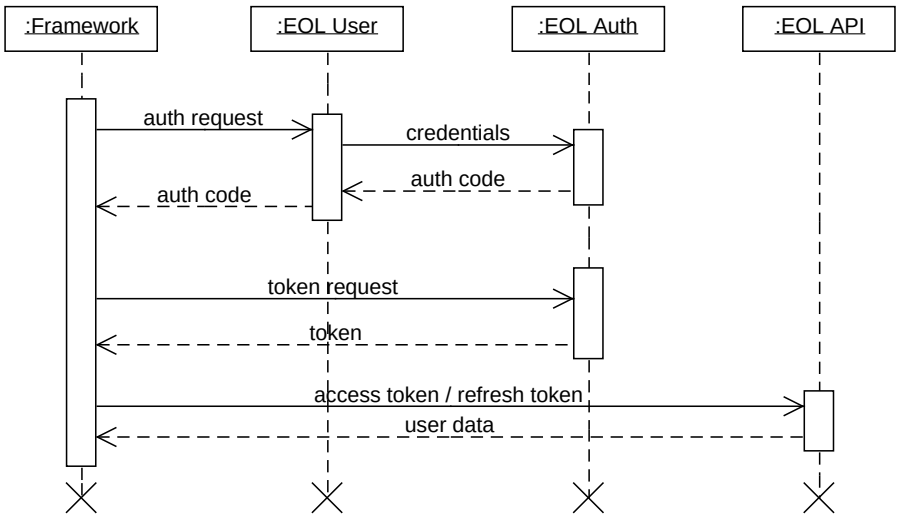


Figure 1: OAuth flow diagram

5.1.4 API Architecture

The architecture of the API needs to be known if the API needs to be extended with additional functionality. Below a short description is given on the API architecture and subcomposition. The Exact Online Web Api can be divided in the following three parts, which are built on each other in the same order.

- .NET Framework functionality
- API Core
- System, Entity and Read; corresponding to different data types.

Two classes are central in the design:

- Dataservice: The DataService class serves as the code endpoint for a data service. It handles data protocol, servide-side paging, authentication, etc.
- Datasource: The DataSource is the main data provider and provides access to all (queryable) data.

5.2 Salesforce

The first and aimed application to integrate the plugins with is Salesforce. These sections give an overview what dependencies exist when writing plugins for Salesforce.

5.2.1 External Authentication Settings

Salesforce provides storage possiblities for developers who want to store authentication settings of users. With the external authentication settings functionality, it is possible to save user-specific authentication data. It might be useful to save the authentication code of a user here, and communicate this to the framework every time data from the API is needed.

5.2.2 Plugin Integration

Usually when writing apps for SalesForce, a lot of rules and limitations are present because of the closed nature of the system. Salesforce uses an own programming language to facilitate the creation of apps for their platform. It is called 'Apex' and is a stongly-typed object-oriented language. Apex enables developers to add business logic to their apps, including button clicks, related record updates, and Visualforce pages. Moreover, the Salesforce frameworks needs to know the structure of the external data. Therefore a external object definition must be defined. Each definition maps to a table that contains the data.

Another option is to render only a HTML view within Salesforce and implement your own code at that point by using Javascript. In this way, an plugin can be embedded in a Salesforce app by using Javascript, but still be maintained by the own development team in the own familiar languages.

6 Architectural Design

This section describes our choices and motivation for the architectural design of the framework. First, the type of language to be used is described, the chosen framework and how the implementation should be tested. Finally, UML diagrams show how the systems will be build up on top of the framework and following what type of architecture.

6.1 Choice of Programming Language

The plugin framework will be an application on its own that will not be part of the Exact Online application. The only correspondence will be the interaction with the Exact Online API, which will provide user data to the framework. Therefore the type of language to use is a relatively open choice.

Typical choices for web-based software are either PHP or .NET, because of the flexibility these languages feature to integrate with HTML. Exact Online is built on .NET, a framework developed by Microsoft. PHP is another language that can be run on almost every web server and features a gentle learning curve. Especially the last argument let us decide to go ahead with PHP as language to use and since our time frame is not really big it is a logical choice to work with a language with which we are already familiar with and therefore this language is our primary choice.

However, since Exact Online is built on the .NET framework and it must be ensured that the application can be maintained in the future by Exact it is not an option to go ahead with PHP as language. Therefore

it is unavoidable to do some extra research on the possiblities with the .NET framework and languages which we will discuss in the section below.

6.2 .NET Languages

C#, Visual Basic.NET and ASP.NET are the primary languages used to program on the .NET framework. This framework is developed by Microsoft and especially suited for development for the web. These languages are syntactically very different, but at the end compile to the same intermediary language. Below we discuss the differences shortly:

- *C#:* Based on C; many similarities with Java; cleaner syntax; multi-line comments; static classes; iterative loops
- *VB.NET:* Based on Basic; syntax more textual; not case-sensitive
- *ASP.NET:* Successor to Microsoft's Active Server Pages (ASP) technology. Dynamically building webpages, same purpose as PHP.

Because of above comparison, we expect C# to be the language we can be familiar with in the shortest time. Since there are almost no performance-differences at runtime, there is no argument for choosing any other language above this one for that reason.

6.3 Testing

Application development will be done using Visual Studio 2013, as also used by other Exact employees. Visual Studio provides good integration with unit tests and let us create a seperate test project next to the main project. Verification and validation will take place to review functionality together with stakeholders.

6.4 High-level Architecture

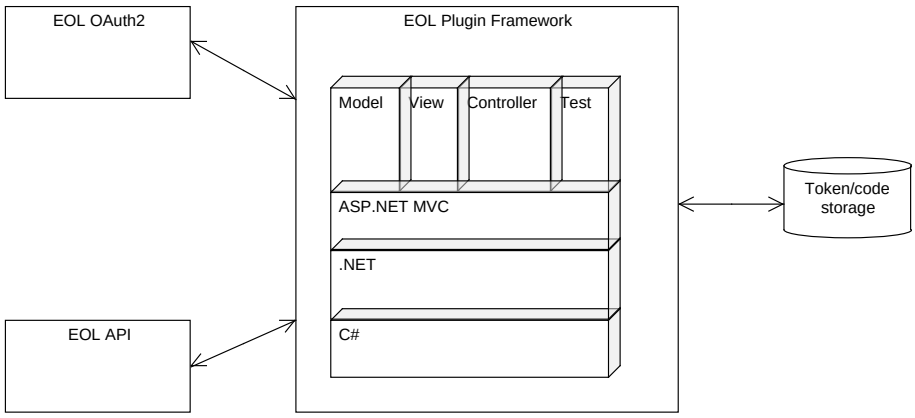


Figure 2: EOL Plugins framework architecture

Seen from a high level, the architecture is built up as follows. The EOL Plugin framework will be built on top of a ASP.NET Model View Controller layer, which is built on top of the .NET libaries, for which we use C# as main programming language.

The framework makes use of two external sources; the Exact API and OAuth authorization in order to obtain the authorization codes and tokens. Once retrieved, the status and codes are stored in a database.

6.5 Class Diagram

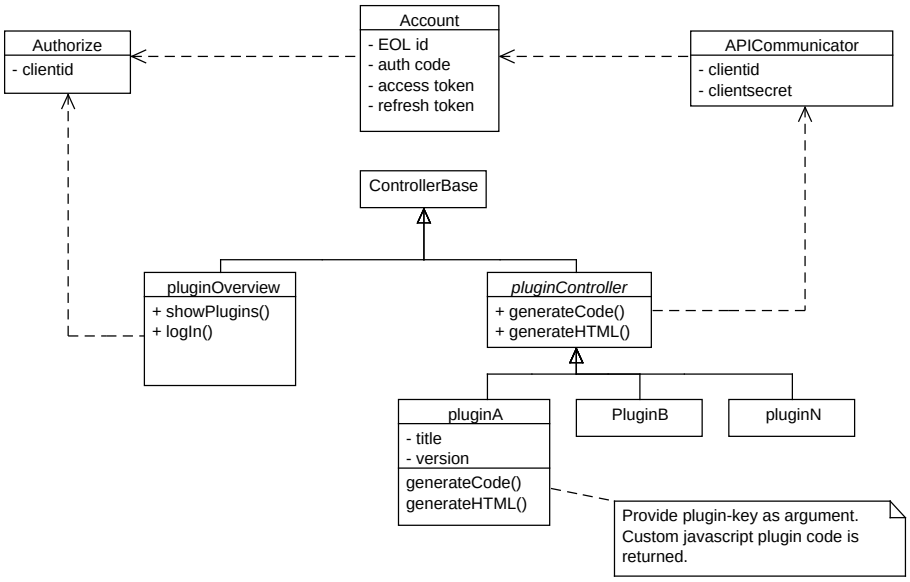


Figure 3: EOL Plugins framework class diagram

Above diagram shows the architecture of the Exact Online plugin framework. The framework is built on top of the .NET and ASP.NET MVC framework and therefore depends the system on some base class provided by .NET. As can be seen on the class diagram every controller inherits its base functionality from the .NET's ControllerBase class. Views are written in ASP.NET, controllers and models in C#.

Basically, the architecture features multiple MVC structures inside a MVC structure. To achieve a high state of flexibility, each plugin features its own model, view and controller so that adding a new plugin does not require changes over the whole system. Every plugin features a controller that inherits all basic plugin functionality from an abstract pluginController class.

Moreover, every plugin features two core functions. A code generator, in order to be able to generate the specific plugin code that Exact Online clients can use, to integrate the plugin within their systems. On the other hand, a generateHTML() class that is used to render the view of the specific plugin.

Every plugin must contain a plugin-key, by which the plugin will identify itself. This key will be integrated in the script. When a plugin requests some information, it has to send this plugin-key to grant access to the personal information. The plugin-code is generated by the pluginController, and differs from the authorization code from the API. This is done to avoid manually updating the key by the plugin. So, the plugin can always use the same code, even when the authorization code changes.

7 Definition of Done

The project can be defined as done when the following goals are reached: Outstanding items plugin is implemented, tested and validated by stakeholders; The system is setup as generic as possible to allow easy extension; as a framework. Argumentation on all design choices is provided in documents. Documents contain the Product plan, Orientation report and Final report following the requirements from Blackboard.

Moreover, for each sprint the project team should make sure that all features are implemented, fully tested, documented and clarified with inline-comments.

Optionally, when the time frame allows, another plugin will be implemented that will show revenue lists. However, providing a highly customizable and feature-rich plugin is preferred above delivering multiple plugins with less functionality.

C Guide on How To Extend

How to extend

This document will describe how to extend the framework with new plugins and this is done by creating a new plugin called 'ExamplePlugin' which will show the contacts with their name and code. First of all, the folder structure is described. Secondly the new models are highlighted. Thirdly the extension of the view is shortly described and finally a new controller is created which is explained in detail.

Folder structure

The EOLPlugin project has an organized structure. Some basic implementation is available that every plugin should use, like PluginControllers for newly created Controllers and AbstractDataRetriever while extending functionality with respect to the API.
Each plugin has its own Controller as shown in Figure 1. In this case the ExamplePluginController will be implemented. For each plugin a folder is created in Models.Plugin which contains classes that are needed for that particular plugin, in this case classes Contact and ExampleRetriever.

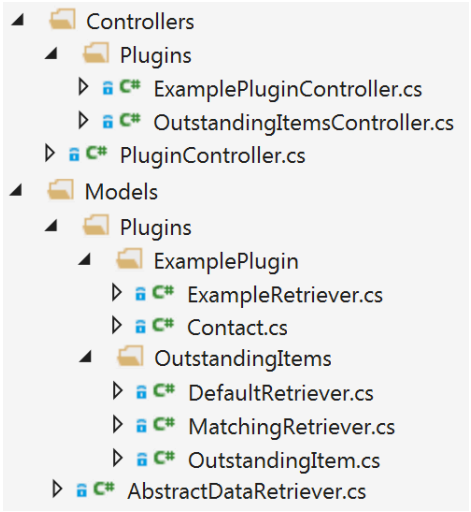


Figure 1: Folder structure

Model

Basic functionality like switching divisions and checking if the user is authorized is already available in AbstractDataRetriever. However, additional functionality is often required when creating a new plugin. This can easily be done by extending AbstractDataRetriever and implement additional methods. AbstractDataRetriever needs an user id so that it can use the right tokens when calling the API.

```
namespace EOLPlugins.Models.Plugins.ExamplePlugin
{
    public class ExampleRetriever : AbstractDataRetriever
    {
        public ExampleRetriever(string user_id) : base(user_id)
        {
        }

        public Contact[] getContacts()
        {
            JsonValue json = api.request("/read/crm/Contacts")
                                .select("Code, FullName")
                                .orderBy("FullName asc")
                                .get();

            Contact[] result = new Contact[json.Count];

            for (int i = 0; i < json.Count; i++)
                result[i] = new Contact(
                                (string) json[i]["Code"],
                                (string) json[i]["FullName"] );

            return result;
        }
    }

    public class Contact
    {
        public string code { get; set; }
        public string name { get; set; }

        public Contact(string code, string name)
        {
            this.code = code;
            this.name = name;
        }
    }
}
```

View

At this moment, the view does not need to be updated when creating a new plugin.

Controller

This section will describe how additional controllers can be created. Suppose an additional plugin 'ExamplePlugin' is needed, then a controller called 'ExamplePluginCon-

troller' has to be added in Controller>Plugins folder and it must extend the PluginController. The PluginController takes care of authorization, user ids and basic information needed in the plugin View (sent via ViewBag). The user id is put into a session variable called "uid". When extending the PluginController, several methods needs to implemented and most of them are self-explanatory. The setRetriever method, however, has one import aspect: it has to set the base.retriever to an DataRetriever object to make sure no nullreferenceexceptions can occur, since it determines if the user is authorized. In addition to this, the retriever takes care of the communication with the API.

```
namespace EOLPlugins.Controllers.Plugins
{
    public class ExamplePluginController : PluginController
    {
        protected new ExampleRetriever retriever;

        public ExamplePluginController()
        {
        }

        protected override string getPluginTitle()
        {
            return "An Example Plugin";
        }

        protected override string[] getColumnNames()
        {
            return new string[] { "Code", "Name" };
        }

        protected override string getNoContentMessage()
        {
            return "Nothing to show right now.";
        }

        protected override void setRetriever()
        {
            this.retriever = new ExampleRetriever((string)Session["uid"]);
            base.retriever = this.retriever;
        }

        public override ActionResult generateView()
        {
            Contact[] contacts = retriever.getContacts();
            string[][] content = new string[contacts.Length][];

            for(int i = 0; i < contacts.Length; i++)
            {
                content[i] = new string[] {
                                    contacts[i].code,
                                    contacts[i].name };
            }

            setTableContent(content);
            return View("Plugin");
        }
    }
}
```

D Guide on How To Implement

This document describes how to publish the Exact Online Plugins system on an Azure website and how to implement plugins built on the Exact Online platform into other environments. Lastly, a description is given on how to maintain and use the prototype of the Salesforce app.

Publish Azure website

To test the prototype by using a secure connection (https), a virtual machine is set up to deploy the prototype on. The system is hosted at Microsoft Azure, which easily integrates with Microsoft Visual Studio. 'Microsoft Azure is an open and flexible cloud platform that enables you to quickly build, deploy and manage applications across a global network of Microsoft-managed data center[2]'.

A project in Visual Studio needs to have a publish profile set up to Azure, so that it knows the location of the project. Since a publish profile is included in the EOLPlugins project, one can easily publish the full project to the Azure website by performing the following operations:
Right click on the EOLPlugins project, then Publish. Changing settings is optional. By clicking again on Publish, the required files are being uploaded to the server and after this is done, a new window will open that shows the webpage.

Implementation Environments

For the system to work in a third-pary application, the third-party application must allow the use of in-line javascript. The piece of Javascript below will create an inline frame (iframe) once a page is loaded, and will load the specific requested plugin inside that iframe.

```
<script language="JavaScript" type="text/javascript">
   plugin = 'OutstandingItems';
   uid = 'some_kind_of_id';
</script>
<script language="JavaScript" type="text/javascript"
        src="EOLPLUGINS_URI/Content/js/EOLplugin.js"></script>
```

The third-party application needs to provide a unique userid, preferably hashed, to identify the user who is requesting a plugin. This user id ('uid') might be static, but might be generated as well. A typical format for the Salesforce application would be 'sf_hashed(salesforce-userid)'.

Salesforce app

Salesforces provides a marketplace (Salesfoce App Exchange [3]) for apps on which developers can publish their apps/plugins. In Salesforce the project team developed a test app, that integrates an Exact Online plugin via Javascript. This code might be used to publish a final version of the app in the App Exchange. Someone interested may follow the following actions to find the source code.

- To view this app, log in as edgar.wieringa@exact.com / Online123
- The code can be modified by clicking 'Edgar Wieringa' (on the top) -> Developer Console

[2] http://azure.microsoft.com/en-us/
[3] https://appexchange.salesforce.com/

Note that before an app is published to the App Exchange, the final url to the system needs to get updated. This url will be the location of the deployment server. This server is required to use a secured http (https) connection, so that iframes will not be blocked by browsers because of security reasons in the case that a third-party application runs https as well.