Report for Enterprise Software Infrastructure Project

Product ordering process automation with process logging in blockchain

Oliver Faust

oliver.faust@students.fhnw.ch

oliversimon.faust@studenti.unicam.it

Carlo Mehli

Carlo.mehli@students.fhnw.ch

carlogiacomo.mehli@studenti.unicam.it

January 28, 2020

Andrea Morichetta

Master of Science in Computer Science

Table of Contents

1. Introduction

When a potential customer interacts with a product configurator website and expresses interest in a product, often a complicated business process will be triggered. Such functionality is often to a large degree invisible to the customer, but it includes various tasks and business logic that has to be modelled and developed. The following described software for a laptop wholesaler company, automates this process wherever possible, without decreasing customer service.

In the following chapters, the process will be introduced and the general goal of the project as well as business case will be described.

1.1 Goal

The overall goal of the project is to provide a collaboration model solving a real case study by implementing the case in Camunda workflow engine. Moreover, a webservice shall be provided in order to solve a specific business problem. This Webservice needs to be integrated in the collaboration model and deliver business value. The collaboration will keep track of process steps and the activities executed by the users. In order to be auditable and to enable later process mining activities, the process logs should be stored to the Ethereum Blockchain using a specific smart contract.

1.2 Business Case

In this particular scenario, the company is a laptop wholesaler selling large quantities of laptops in a b2b market environment. The company provides a website with an online product configurator. The potential customer (lead) can use this configurator to request a product offer. Even if the customer does not complete the configuration of the product, by sending an request for an offer, the customer information will still be registered as a lead within the CRM (Bexio). If the configuration is completed, the sales team will contact the customer (if requested by the customer) and evaluate the offer request. A price will be calculated depending on parameters such as customer type and number of items requested. Furthermore, a risk assessment of the offer will be performed to assess, if the offer is of great importance or the offer is likely to be declined and therefore should be checked manually by a supervising salesman. The customer interactions will be stored in an external CRM tool (Bexio). Moreover, an offer will be generated in the CRM according to the parameters of the configurator and the price calculation. This offer will be sent to the customer by e-Mail. If the customer accepts the offer using a specific link, the offer will be turned into an order and will be further processed. In case the customer does not answer within two weeks or declines the offer, a sales employee will contact the customer and will discuss the offer with the customer. If needed, the current offer is adapted by the salesman and sent again to the customer. As soon as the customer accepts the offer, an invoice will be generated by the CRM and the good will be prepared and shipped within logistics department. After the good has been shipped, an invoice will be sent to the customer. Each step of the process will be logged to the Etherum Blockchain, in order to allow Process Mining and log the process execution.

2. Process Model

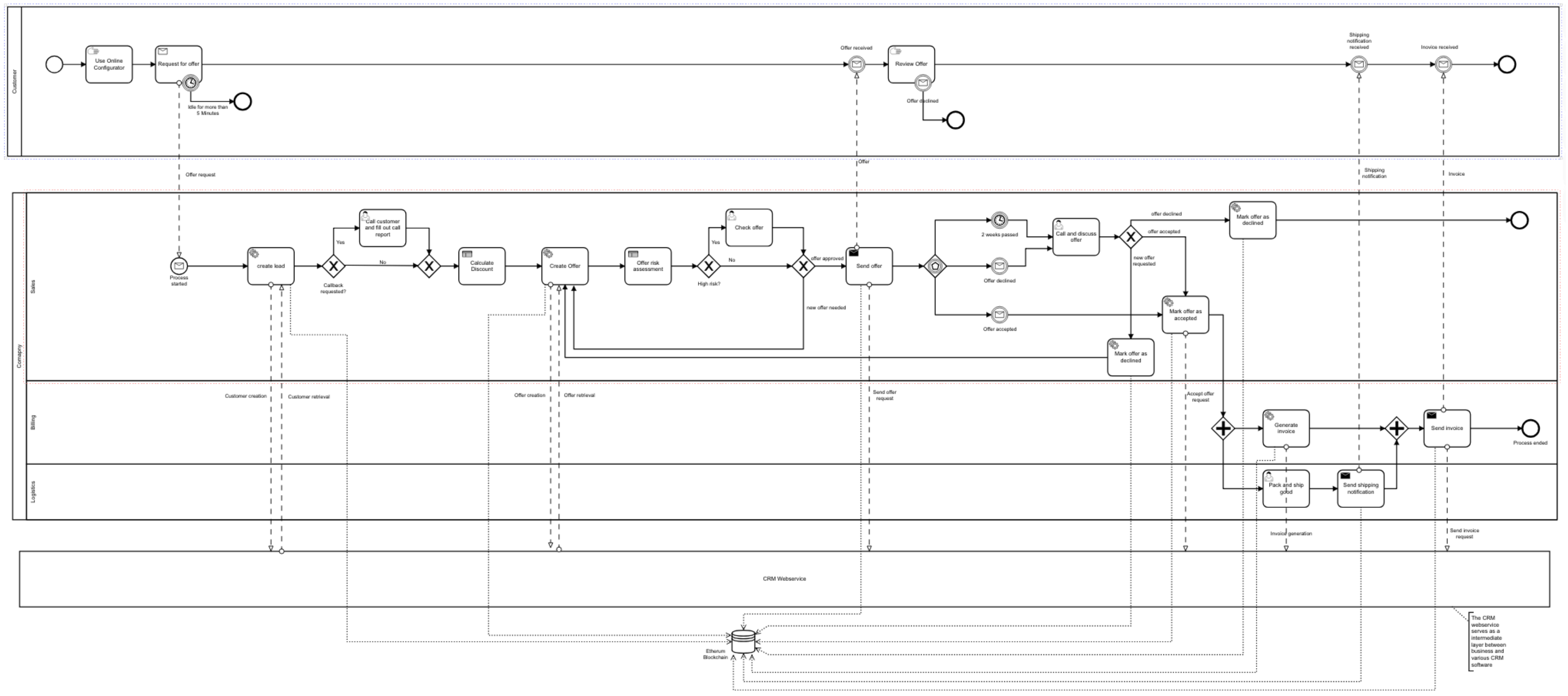
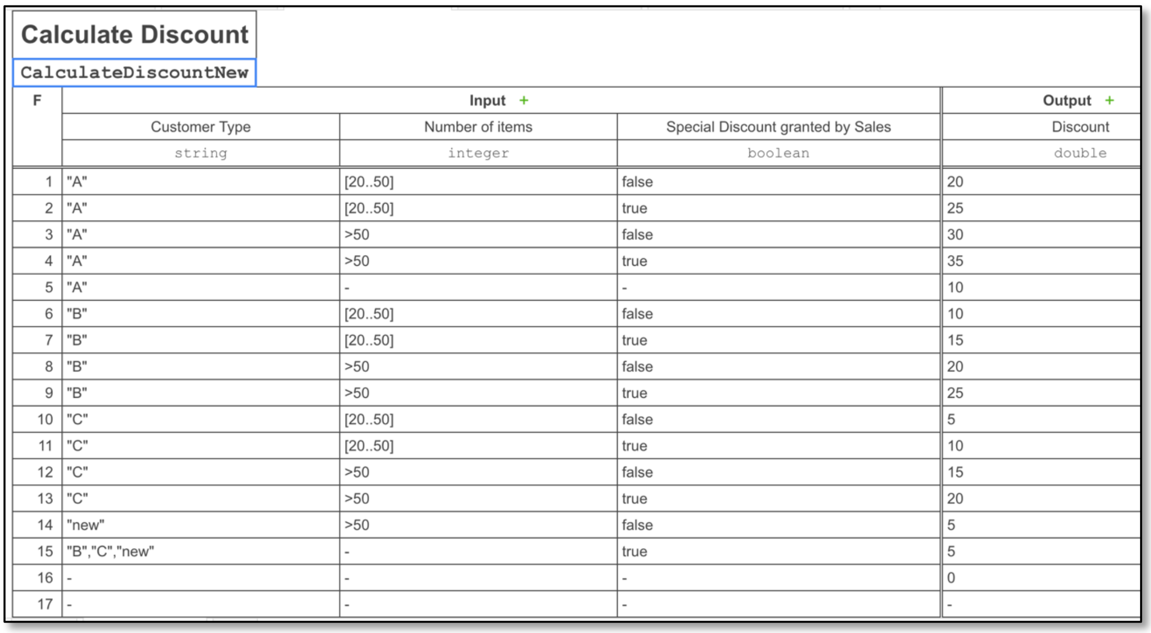


Figure 1: Product Ordering Process as BPMN

3. Process Model Description

- For the company, the process begins as soon as the potential customer has used the product configurator and requests an offer.
- The customer's data are then sent to the webservice which serves as intermediate layer between CRM and the business logic. In case of an existing customer, the customer data is retrieved from the CRM using the Webservice. Otherwise, a new Lead is being created.
- In case the customer requested a call by the salesman, a human task with all the customer data is assigned to the salesman. After the call he can update the form with the information gathered from the customer and close the task.
- According to the customer & configuration data, the discount is automatically calculated according to the following decision table:



Calculate Discount

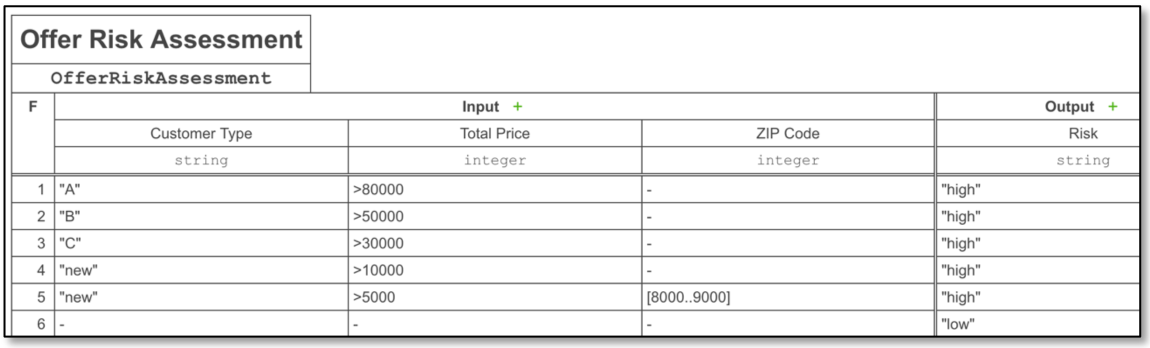
CalculateDiscountNew

F	Input +			Output +
	Customer Type	Number of items	Special Discount granted by Sales	Discount
	string	integer	boolean	double
1	"A"	[20..50]	false	20
2	"A"	[20..50]	true	25
3	"A"	>50	false	30
4	"A"	>50	true	35
5	"A"	-	-	10
6	"B"	[20..50]	false	10
7	"B"	[20..50]	true	15
8	"B"	>50	false	20
9	"B"	>50	true	25
10	"C"	[20..50]	false	5
11	"C"	[20..50]	true	10
12	"C"	>50	false	15
13	"C"	>50	true	20
14	"new"	>50	false	5
15	"B","C","new"	-	true	5
16	-	-	-	0
17	-	-	-	-

Figure 2: DMN to determine customer discount

- The calculated discount together with the ordered product information is sent to the CRM Webservice. The CRM will then create an offer accordingly.

- In order to assess whether a salesman should check the offer (in case the offer is at risk to be declined, or the offer is highly important for the company), an automatic risk assessment is done, according to the following decision table.



Offer Risk Assessment

OfferRiskAssessment

F	Input +			Output +
	Customer Type	Total Price	ZIP Code	Risk
	string	integer	integer	string
1	"A"	>80000	-	"high"
2	"B"	>50000	-	"high"
3	"C"	>30000	-	"high"
4	"new"	>10000	-	"high"
5	"new"	>5000	[8000..9000]	"high"
6	-	-	-	"low"

Figure 3: DMN to assess the risk of an offer

- If the risk is assessed as high, a human task will be assigned to the salesman in order to check, and if necessary, adapt the parameters (price, discount, customer data) of the offer. If changes are made, the offer will be newly generated with the new parameters.
- The offer is then sent to the customer by e-Mail. The customer receives a hyperlink to access the offer on the website and accept or decline the offer online.
- In case the customer declines the offer, or if he does not react to the offer within two weeks, a human task for the salesman is created containing the customer and offer information, in order to call the client and discuss the offer.
    - If the customer declines the offer during the call, the salesman will mark the offer as declined. Thereafter, this information is sent to the Webservice which marks the offer as declined within the CRM.
    - If the customer requests a new offer, the salesman adapts the parameter of the offer. The information is sent to the Webservice which marks the current offer within the CRM as declined and requests a new offer with the new parameters which is again automatically sent to the customer again.
    - If the customer accepts the offer after the call, the salesman marks the offer as accepted and the process will continue, as if the customer had accepted the offer by himself.

- In case the customer accepts the offer, the offer is marked as accepted within the CRM system and the process moves on to the billing and logistics department, where tasks are executed in parallel.
- In the billing department the invoice is automatically created according to the accepted offer. The invoice is created by the CRM.
- The logistics department packs and ships the good. As soon as the good is shipped, the logistics employee, marks the goods as shipped within the system, and a notification will be sent to the customer.
- The invoice is sent, as soon as the shipping department has completed its tasks. The invoice is sent to the customer's email address.

4. Components

In order for the process to be started, human tasks to be executed, or invoices/offers to be created, several frontends are used by the customer and employees. The front ends and their function within the process will be described following.

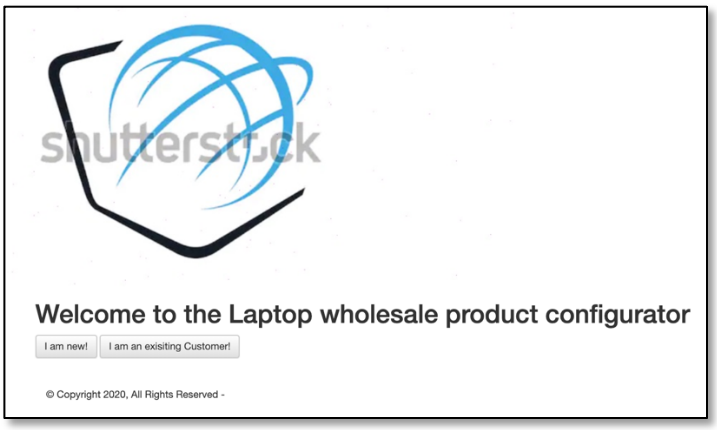
4.1 Product Configurator Website

The website is used by the customer in order to configure the desired product as well as review the offer. To start the configurator, and hence the whole process go to: http://localhost:8080/ProductConfiguratorWebsite/WebContent.

The website uses JavaScript to interact with the Camunda REST-API. Hereafter, the used JavaScript structure of the website is described:

- camundaManager.js
  - This script handles all generic interactions with the Camunda REST-API. It implements generic functions to start a Camunda Process and to send messages.
- processFlowController.js
  - This script contains functionality to steer the flow within the website for the use of the product configurator.
- offerController.js
  - This script handles the interaction when the user receives a link with the offer. It fetches the query strings and interact with Camunda via REST messages.
- Utils.js
  - Provides utility functions to send Ajax Requests and generate UUIDs.

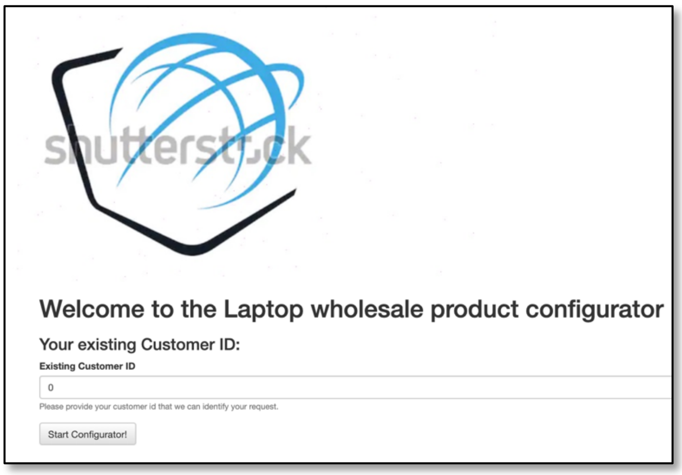
In order to use the product configurator, the client needs to sign up (new customer) or sign in using an existing customer id. (Note that for this project, security related aspects have not been considered). Therefore, the customer initially needs to choose whether he is an existing customer or not.


Welcome to the Laptop wholesale product configurator

I am new!
I am an exisiting Customer!
© Copyright 2020, All Rights Reserved -

Figure 4: Welcome page of product configurator

As an existing customer, only the customer ID needs to be entered in order to use the product configurator.


Welcome to the Laptop wholesale product configurator

Your existing Customer ID:

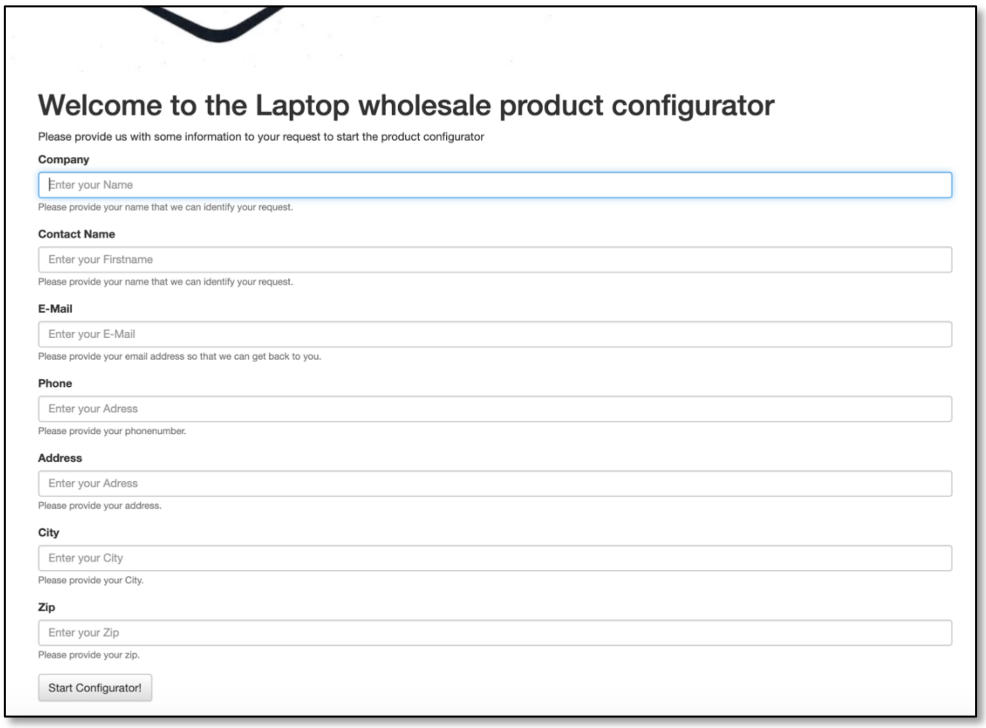
Existing Customer ID

0

Please provide your customer id that we can identify your request.

Figure 5: Page for existing customers

As a new customer, the basic informations such as name, address, phone number need to be entered.


Welcome to the Laptop wholesale product configurator

Please provide us with some information to your request to start the product configurator

Company	Enter your Name
	Please provide your name that we can identify your request.
Contact Name	Enter your Firstname
	Please provide your name that we can identify your request.
E-Mail	Enter your E-Mail
	Please provide your email address so that we can get back to you.
Phone	Enter your Adress
	Please provide your phonenumber.
Address	Enter your Adress
	Please provide your address.
City	Enter your City
	Please provide your City.
Zip	Enter your Zip
	Please provide your zip.

Figure 6: Website for new customers

As soon as the customer is registered, he can configure the laptop as desired and can choose the quantity of the product. Please note, that the configurator is targeted at retailers, and business customer who order large amounts of the same product. In case the customer wishes to discuss any questions with a sales employee, he has the opportunity to request a callback.

On click of the “start configurator” button, the websites interacts with the Camunda REST-API and the customer process will be started. First however, it fetches the current process definition.

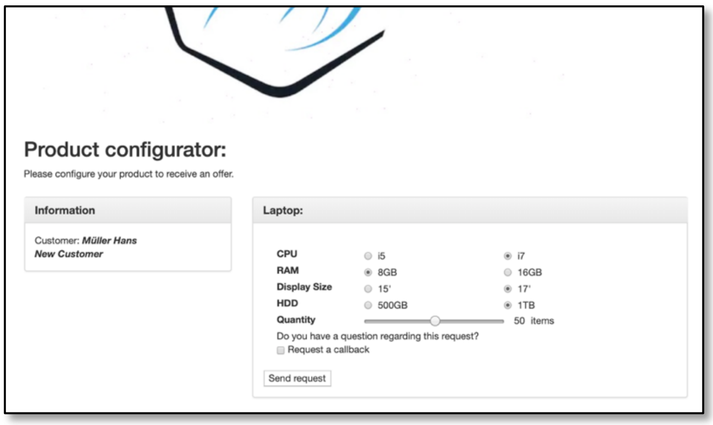
Fetch current process definition:

```
GET http://localhost:8080/engine-rest/process-
definition?key=ProductConfiguratorProcessCustomer&latestVersion=true
/start
```

Start process:

```
POST http://localhost:8080/engine-rest/process-definition/2cba8c23-36cd-11ea-
a1e8-dae16e121990/start

{"businessKey":"8547138a-1505-8fef-b336-84781717a08a"}
```



Product configurator:

Please configure your product to receive an offer.

Information

Customer: ***Müller Hans***
New Customer

Laptop:

CPU	i5	[✓ i7]
RAM	[✓ 8GB]	16GB
Display Size	15'	[✓ 17']
HDD	500GB	[✓ 1TB]
Quantity		50 items

Do you have a question regarding this request?
☐ Request a callback

Figure 7: Product configurator for new and existing customers

After the configuration has been finished by clicking on the "send request" button, a REST call to the message endpoint will be sent in order to initiate the process on the companies side. The website will generate a unique identifier (Business Key) which is used throughout the whole process, in order to assure the correct assignment of information exchanged between process instances and the CRM.

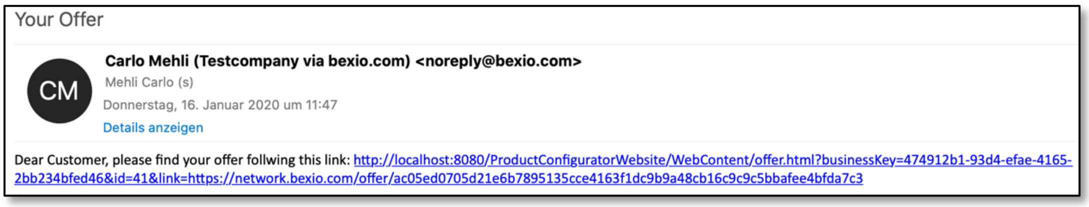
Message to indicate that the configurator has been used and to start the company process:

```
POST http://localhost:8080/engine-rest/message

{
"messageName":"info_req",
"businessKey":"8547138a-1505-8fef-b336-84781717a08a",
"processInstanceId":"bc9c82e9-3928-11ea-a576-dae16e121990",
"all":true,
"processVariables":
 {  "v_customerName":{"value":"","type":"String"},
    “v_customerFirstName":{"value":"","type":"String"},
    "v_customerEmail":{"value":"","type":"String"},
    "v_customerPhone":{"value":"","type":"String"},
    "v_customerAddress":{"value":"","type":"String"},
    "v_customerZip":{"value":"","type":"Integer"},
    "v_customerCity":{"value":"","type":"String"},
    "v_customerID":{"value":"4","type":"Integer"},
    "v_productDesc":
{"value":"Premium Laptop: i7 8gb RAM 1TB SSD 17inch Monitor","type":"String"},
    "v_productAmount":{"value":"53","type":"Long"},
    "v_productPrice":{"value":1550,"type":"Long"},
    "callbackRequested":{"value":false,"type":"Boolean"}
 }
}
```

4.2 Offer acceptance & invoice website

When the offer is sent to the customer, he receives a link which points to the offer acceptance website. In the query string of the link, all necessary information is encoded. This information is handled by offerController.js.



Your Offer

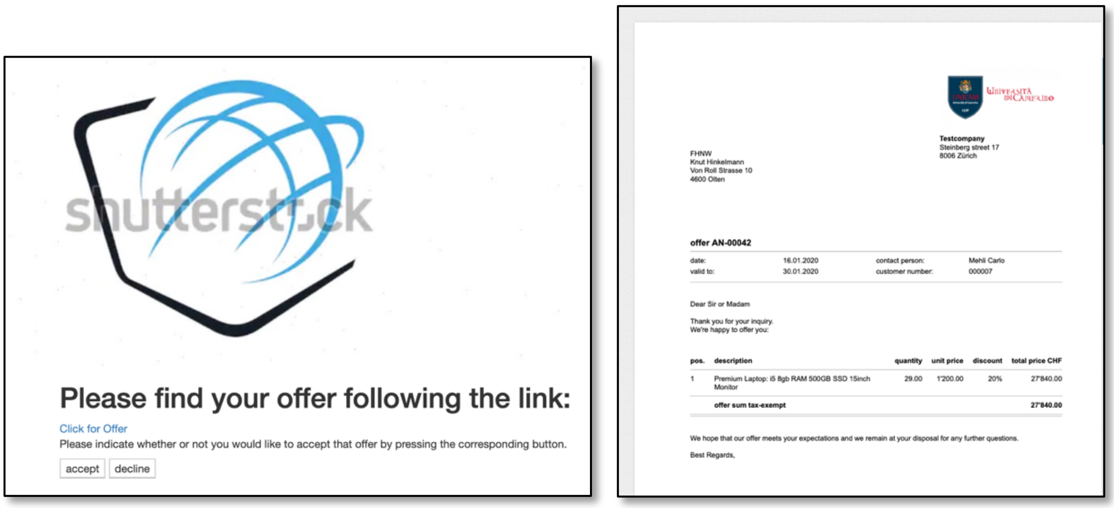
CM

Carlo Mehli (Testcompany via bexio.com) <noreply@bexio.com>
Mehli Carlo (s)
Donnerstag, 16. Januar 2020 um 11:47
Details anzeigen

Dear Customer, please find your offer follwing this link: http://localhost:8080/ProductConfiguratorWebsite/WebContent/offer.html?businessKey=474912b1-93d4-efae-4165-2bb234bfed46&id=41&link=https://network.bexio.com/offer/ac05ed0705d21e6b7895135cce4163f1dc9b9a48cb16c9c9c5bbafee4bfda7c3

Figure 8: Example of an e-Mail containing the link to the acceptance offer website

On the website, the customer is able to access the offer by clicking on the link. The customer can access the offer directly within the CRM. This way we assure, that the customer always accesses the newest version of the offer. Furthermore, the customer can directly accept or deny the offer.



Please find your offer following the link:

Click for Offer

Please indicate whether or not you would like to accept that offer by pressing the corresponding button.

accept decline

Testcompany
Steinberg street 17
8006 Zürich

FHNW
Knut Hinkelmann
Von Roll Strasse 10
4600 Olten

offer AN-00042

date:	16.01.2020	contact person:	Mehli Carlo
valid to:	30.01.2020	customer number:	000007

Dear Sir or Madam

Thank you for your inquiry.
We're happy to offer you:

pos.	description	quantity	unit price	discount	total price CHF
1	Premium Laptop: i5 8gb RAM 500GB SSD 15inch Monitor	29.00	1'200.00	20%	27'840.00
	offer sum tax-exempt				**27'840.00**

We hope that our offer meets your expectations and we remain at your disposal for any further questions.

Best Regards,

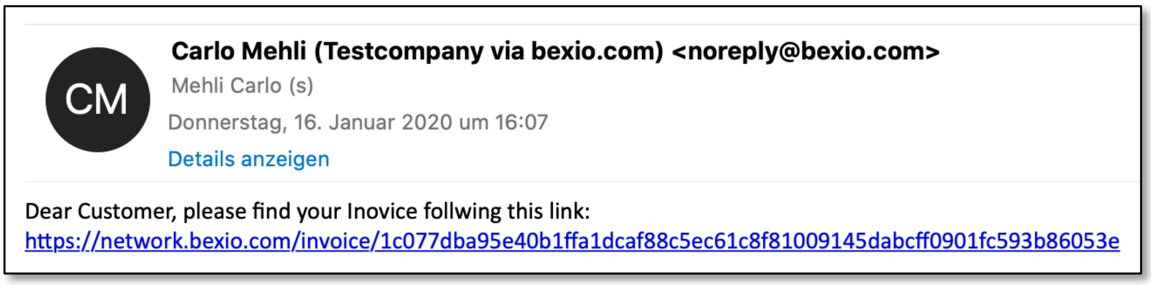
Figure 9: Example of the acceptance website including a correspondent product offer

Message to inform Camunda about customers decision regarding the offer:

```
POST http://localhost:8080/engine-rest/message

{"messageName":"offer_accepted",
"businessKey":"8547138a-1505-8fef-b336-84781717a08a",
"all":true,
"processVariables":{"v_offerID":{"value":"45","type":"Integer"}
}
}
```

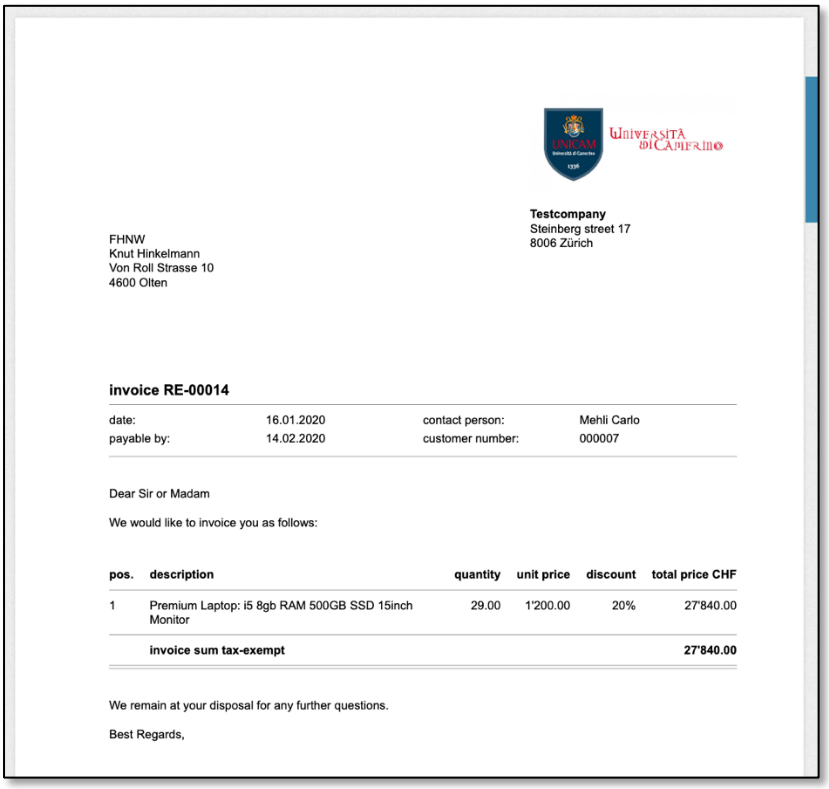
If the offer is accepted, the invoice will be sent to the customer in the same manner. The customer can access the invoice directly in the CRM by accessing the hyperlink sent by e-Mail.



Carlo Mehli (Testcompany via bexio.com) <noreply@bexio.com>
Mehli Carlo (s)
Donnerstag, 16. Januar 2020 um 16:07
Details anzeigen

Dear Customer, please find your Inovice follwing this link:
https://network.bexio.com/invoice/1c077dba95e40b1ffa1dcaf88c5ec61c8f81009145dabcff0901fc593b86053e

Figure 10: Example of an invoice sent by e-Mail



Testcompany
Steinberg street 17
8006 Zürich

FHNW
Knut Hinkelmann
Von Roll Strasse 10
4600 Olten

invoice RE-00014

date:	16.01.2020	contact person:	Mehli Carlo
payable by:	14.02.2020	customer number:	000007

Dear Sir or Madam

We would like to invoice you as follows:

pos.	description	quantity	unit price	discount	total price CHF
1	Premium Laptop: i5 8gb RAM 500GB SSD 15inch Monitor	29.00	1'200.00	20%	27'840.00
	invoice sum tax-exempt				**27'840.00**

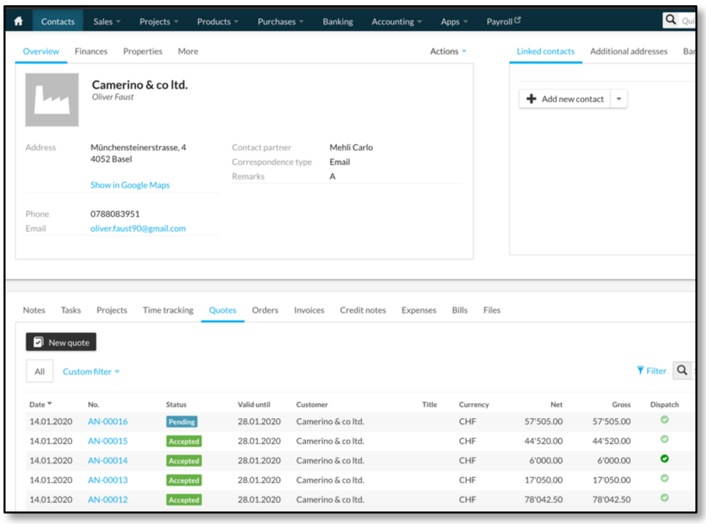
We remain at your disposal for any further questions.

Best Regards,

Figure 11: Example of an invoice accessed by the customer

4.3 CRM Bexio

Within this project, the used CRM tool is called Bexio[1]. However, thanks to the chosen software architecture and the programmed Webservice, any CRM Tool could be used for the process (See 5 Webservice). Within Bexio the customers are called "Contacts" and offers are called "Quotes". For each contact one can see all the quotes and if they have already been accepted. Of course, also the invoices and their status can be retrieved for each contact. The customer type is assigned using the "Remark"-field within Bexio. Changes to customers, invoices or offers could also be done directly within Bexio.



Camerino & co ltd.
Oliver Faust

Address	Münchensteinerstrasse, 4 4052 Basel
	Show in Google Maps
Phone	0788083951
Email	oliver.faust90@gmail.com
Contact partner	Mehli Carlo
Correspondence type	Email
Remarks	A

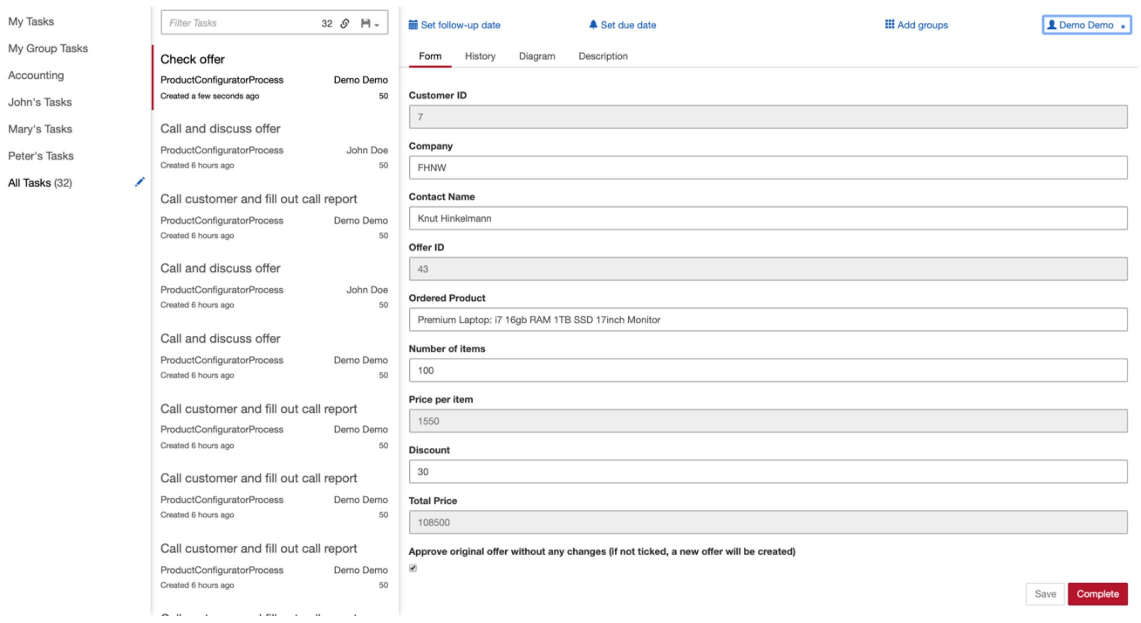
Date	No.	Status	Valid until	Customer	Title	Currency	Net	Gross	Dispatch
14.01.2020	AN-00016	Pending	28.01.2020	Camerino & co ltd.		CHF	57'505.00	57'505.00	✓
14.01.2020	AN-00015	Accepted	28.01.2020	Camerino & co ltd.		CHF	44'520.00	44'520.00	✓
14.01.2020	AN-00014	Accepted	28.01.2020	Camerino & co ltd.		CHF	6'000.00	6'000.00	✓
14.01.2020	AN-00013	Accepted	28.01.2020	Camerino & co ltd.		CHF	17'050.00	17'050.00	✓
14.01.2020	AN-00012	Accepted	28.01.2020	Camerino & co ltd.		CHF	78'042.50	78'042.50	✓

Figure 12: Customer data within the CRM including the corresponding offers

[1] https://www.bexio.com/en-CH/

4.4 Camunda Tasklist

The process involves some manual tasks which have to be executed by the sales or logistic employee. These manual tasks are primarily related to risk management and personal contact with the customer. The tasks are automatically assigned to the correct employee. In our example the sales employee is "John". He is responsible for callbacks (if requested by the client) or calls if the customer declines or does not respond to an offer. "Mary" is the supervisor of "John" and therefore checks only offers which are assessed with high risk. All logistic tasks will be assigned to "Peter", who is responsible for packing and shipping. Hereafter, an excerpt of the manual tasks are shown:



Check offer
ProductConfiguratorProcess — Demo Demo
Created a few seconds ago — 50

Call and discuss offer
ProductConfiguratorProcess — John Doe
Created 6 hours ago — 50

Call customer and fill out call report
ProductConfiguratorProcess — Demo Demo
Created 6 hours ago — 50

Call and discuss offer
ProductConfiguratorProcess — John Doe
Created 6 hours ago — 50

Call and discuss offer
ProductConfiguratorProcess — Demo Demo
Created 6 hours ago — 50

Call customer and fill out call report
ProductConfiguratorProcess — Demo Demo
Created 6 hours ago — 50

Call customer and fill out call report
ProductConfiguratorProcess — Demo Demo
Created 6 hours ago — 50

Call customer and fill out call report
ProductConfiguratorProcess — Demo Demo
Created 6 hours ago — 50

Customer ID	7
Company	FHNW
Contact Name	Knut Hinkelmann
Offer ID	43
Ordered Product	Premium Laptop: i7 16gb RAM 1TB SSD 17inch Monitor
Number of items	100
Price per item	1550
Discount	30
Total Price	108500
Approve original offer without any changes (if not ticked, a new offer will be created)	✓

Figure 13: Example of a human task to check the offer

5. Webservice

The Webservice provides CRM functionality to calling business services. It serves as an intermediate layer **(Broker pattern)** between business processes and application and various CRM tools. Due to its architecture, a very loose coupling is achieved between calling applications and the CRM. This is done through the **Factory Design Pattern**, as well as an intermediate layer of **business data (transfer/exchange) objects** which define the semantic meaning of business documents, such as customers or offers. Every class which implements the CRMConnector interface can be used as a connector to a CRM solution. Since it is most likely that various CRM solutions will store customer information in a different data format as well as will have a slightly different terminology, each connector must provide mapping functionality to so called Exchange objects. Such objects (e.g. CustomerExchange) are POJOs containing all data fields the business wants to use. Moreover, the Exchange object are used as exchange data structures for the webservice. Calling applications will hence not have to worry about what kind of data structure various CRM-Tools will need. Thanks to this webservice, the company achieves a loose cupelling against the used CRM-Tool and could easily change the CRM in the future without any need to change code in calling applications throughout its business processes.

5.1 Architecture

As can be seen in Figure 15: CRMConnector Factory Pattern UML Class diagramFigure 15, the central element is the CRMConnector interface. It defines all the functionality an implementing CRM connector must have. In our example, a connector for the CRM "Bexio" was implemented. The REST-API endpoints use the CRMConnectorFactory to get a particular CRMConnector and forward requests to the implementing Connector. The BexioConnector itself basically just consumes the REST-API of Bexio[2].

[2] For more information about the Bexio API see: https://docs.bexio.com/legacy/index.html. For the sake of simplicity, we use the now deprecated signature based authentication mechanism which has been replaced by oAuth2.

Figure 14 illustrates the design of the POJOs. Business object are modelled in the Exchange Classes. Respective CRM data object are modelled accordingly. However, all POJOs must implement the respective marker interface (Eg. Customer / Order) . The mapping between both data structures must be handled in the respective CRMConnector implementation.

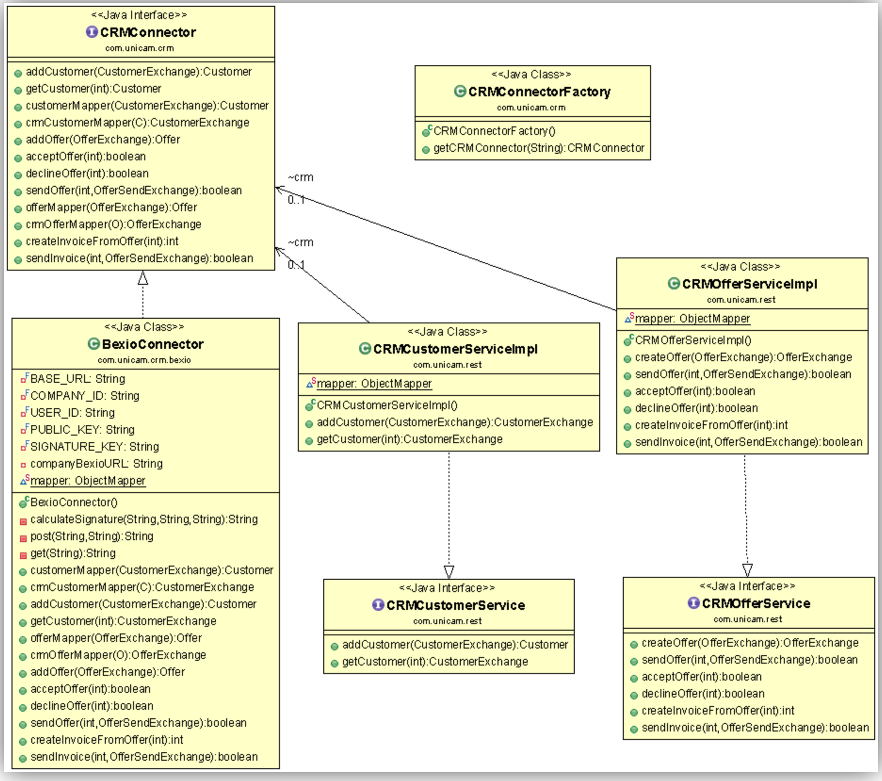

<<Java Interface>>
CRMConnector
com.unicam.crm
addCustomer(CustomerExchange):Customer
getCustomer(int):Customer
customerMapper(CustomerExchange):Customer
crmCustomerMapper(C):CustomerExchange
addOffer(OfferExchange):Offer
acceptOffer(int):boolean
declineOffer(int):boolean
sendOffer(int,OfferSendExchange):boolean
offerMapper(OfferExchange):Offer
crmOfferMapper(O):OfferExchange
createInvoiceFromOffer(int):int
sendInvoice(int,OfferSendExchange):boolean
<<Java Class>>
CRMConnectorFactory
com.unicam.crm
CRMConnectorFactory()
getCRMConnector(String):CRMConnector
~crm
0..1
~crm
0..1
<<Java Class>>
BexioConnector
com.unicam.crm.bexio
BASE_URL: String
COMPANY_ID: String
USER_ID: String
PUBLIC_KEY: String
SIGNATURE_KEY: String
companyBexioURL: String
mapper: ObjectMapper
BexioConnector()
calculateSignature(String,String,String):String
post(String,String):String
get(String):String
customerMapper(CustomerExchange):Customer
crmCustomerMapper(C):CustomerExchange
addCustomer(CustomerExchange):Customer
getCustomer(int):CustomerExchange
offerMapper(OfferExchange):Offer
crmOfferMapper(O):OfferExchange
addOffer(OfferExchange):Offer
acceptOffer(int):boolean
declineOffer(int):boolean
sendOffer(int,OfferSendExchange):boolean
createInvoiceFromOffer(int):int
sendInvoice(int,OfferSendExchange):boolean
<<Java Class>>
CRMCustomerServiceImpl
com.unicam.rest
mapper: ObjectMapper
CRMCustomerServiceImpl()
addCustomer(CustomerExchange):CustomerExchange
getCustomer(int):CustomerExchange
<<Java Class>>
CRMOfferServiceImpl
com.unicam.rest
mapper: ObjectMapper
CRMOfferServiceImpl()
createOffer(OfferExchange):OfferExchange
sendOffer(int,OfferSendExchange):boolean
acceptOffer(int):boolean
declineOffer(int):boolean
createInvoiceFromOffer(int):int
sendInvoice(int,OfferSendExchange):boolean
<<Java Interface>>
CRMCustomerService
com.unicam.rest
addCustomer(CustomerExchange):Customer
getCustomer(int):CustomerExchange
<<Java Interface>>
CRMOfferService
com.unicam.rest
createOffer(OfferExchange):OfferExchange
sendOffer(int,OfferSendExchange):boolean
acceptOffer(int):boolean
declineOffer(int):boolean
createInvoiceFromOffer(int):int
sendInvoice(int,OfferSendExchange):boolean

Figure 15: CRMConnector Factory Pattern UML Class diagram

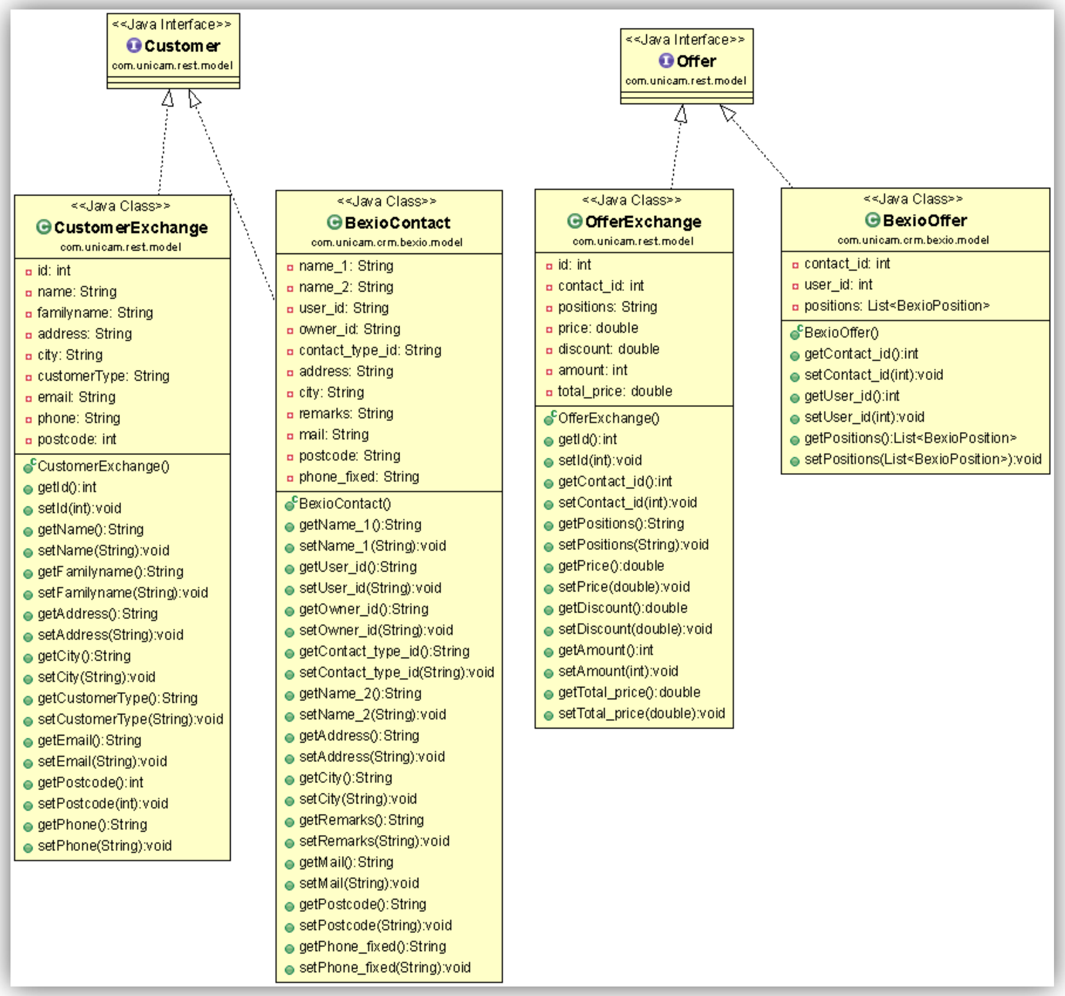


Figure 14: POJOs UML Class diagram

5.2 Services

The Webservice provides the following endpoints, respectively services, to the business:

Customer Management	Order Management
• *Add a Lead (Customer)* • *Get a Lead (Customer)*	• *Create Offer* • *Accept Offer* • *Decline Offer* • *Send Offer via Email* • *Create Invoice from Offer* • *Send Invoice via Email*

The mentioned service which are provided by the Webservice are internally transformed such that they can be used with the CRM. Regarding the Bexio CRM we used the following REST-Resources:

Contact:	https://docs.bexio.com/legacy/resources/contact/
Quotes:	https://docs.bexio.com/legacy/resources/kb_offer/
Invoice:	https://docs.bexio.com/legacy/resources/kb_invoice/

A detailed description of the developed Webservice's API is available in the appendix.

6. Blockchain

During the whole process, the status of the process and the employee executing a specific task, will be logged to an Etherum Blockchain. Therefore a Smart Contract was developed, using solidity. At each step of the process, the information about the status of the process is transformed into to a transaction and saved to the Smart Contract with the following address:

0x07560a88e5F17252a236F47eBC00D6eb7F6dfFB8

The Smart Contract written in solidity, consists of the following information:

```
pragma solidity >=0.4.22 <0.6.0;

contract CamundaProcess {

    struct Task {
        string businessKey;
        string name;
        string executor;
        string additionalInfo;


    }

    mapping(bytes32 => Task[]) private instances;

    function createCollaboration(string memory businessKey) public view returns (bytes32 instanceID){
        instanceID = keccak256(abi.encode(businessKey, block.timestamp));
        return instanceID;
    }

    function registerActivity(bytes32 instanceID, string memory businessKey, string memory taskName, string memory executor, string memory additionalInfo) public{
        instances[instanceID].push(Task(businessKey ,taskName, executor, additionalInfo));
    }

}
```

Within the Java Project, a new BlockchainHelper class was created, which manages the logging to the Blockchain. The BlockchainHelper contains of information such as private Key of the Etherum wallet, Smart Contract address and server address where the Blockchain node is running. We used infura as a HTTPS-Server. The BaseDelegate Class, which all other delegate classes inherit from, instanciates the blockchain-helper object, such that it is easy to use its functionality from the other delegate classes. The methods to register the process activity as a transaction within the Smart Contract, get the private key and create a collaboration are defined within the BlockchainHelper class. Each service task will call the method to register the activity with the respective process information, when it is executed. The transaction hash from every activity is stored as an execution variable in camunda.

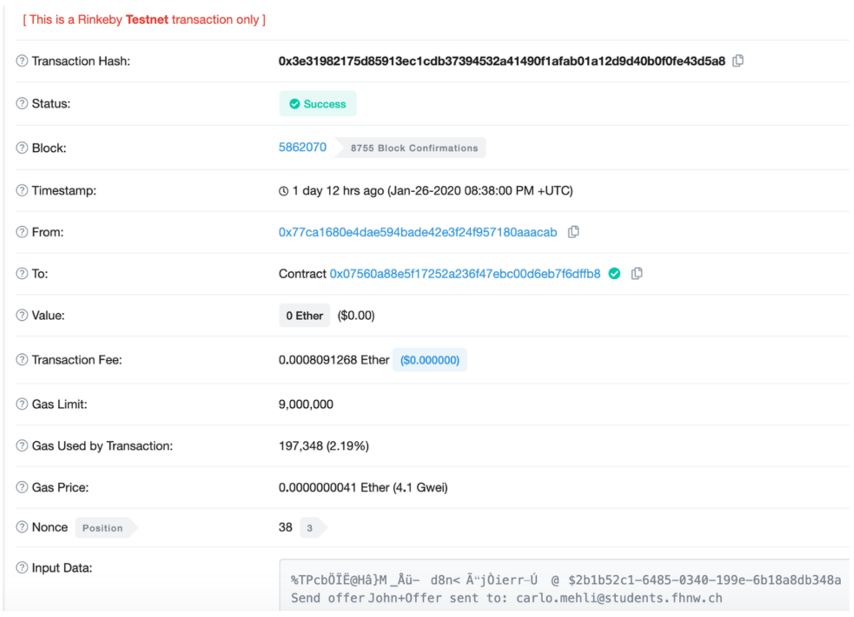
```java
String txHash = blockchain.registerActivity(instanceID, execution.getBusinessKey(), execution.getCurrentActivityName(),"John", "Customer created with id: "+ execution.getVariable("customerID"));

execution.setVariable("TXHASH_createLead", txHash);
```

The logged transactions can be accessed using the following link:

https://rinkeby.etherscan.io/address/0x07560a88e5f17252a236f47ebc00d6eb7f6dffb8

In order to read the details of a transaction, the Input Data format has to be changed to UTF 8:



[This is a Rinkeby **Testnet** transaction only]

Transaction Hash:	0x3e31982175d85913ec1cdb37394532a41490f1afab01a12d9d40b0f0fe43d5a8
Status:	Success
Block:	5862070 8755 Block Confirmations
Timestamp:	1 day 12 hrs ago (Jan-26-2020 08:38:00 PM +UTC)
From:	0x77ca1680e4dae594bade42e3f24f957180aaacab
To:	Contract 0x07560a88e5f17252a236f47ebc00d6eb7f6dffb8
Value:	0 Ether ($0.00)
Transaction Fee:	0.0008091268 Ether ($0.000000)
Gas Limit:	9,000,000
Gas Used by Transaction:	197,348 (2.19%)
Gas Price:	0.0000000041 Ether (4.1 Gwei)
Nonce Position	38 3
Input Data:	%TPcbÖÏË@Hâ}M_Åü- d8n< Ã"jÒierr-Ú @ $2b1b52c1-6485-0340-199e-6b18a8db348a Send offer John+Offer sent to: carlo.mehli@students.fhnw.ch

Figure 16: Example of a transaction within the Smart Contract

For each activity, the following information will be logged:

1. Process unique ID (BusinessKey)
2. Acitivity name (e.g. Send offer)
3. Username, the task is assigned to (e.g. John)
4. Additional information of the activity performed by the service task (e.g. Offer sent to …)

7. Conclusion

The described digitalized process enables the company to reduce human interactions and automate decision making. Moreover, it enables the company to become more auditable thanks to the documented process which can be seen as a single point of truth, as well as the process logging to the blockchain. By making use of design patterns resp. adding an additional layer between the CRM and the various business processes with a CRM-Webservice, we achieve a looser coupling and reduce dependency from the external CRM provider.

Due to the mentioned facts, we can significantly reduce costs, generate business value and improve customer experience .

List of Figures / Tables

Figures

Appendix / Appendices

Source Code:

https://github.com/ofaust90/SPM-Project

Product Configurator Website:

http://localhost:8080/ProductConfiguratorWebsite/WebContent

Camunda Tasklist:

http://localhost:8080/camunda/app/tasklist/default/#/login

Credentials:

User: demo

Password: demo

Bexio Credentials:

Bexio CRM can be access online via www.bexio.com

Credentials:

User: carlo.mehli@students.fhnw.ch

Password: Camerino2019

Please note that this particular Bexio instance is on a trial account (30 Days) and only available until 14.02.2020! If you want to access it later please contact us in order to setup a new trial account.

Blockchain

Contract Address:

0x07560a88e5f17252a236f47ebc00d6eb7f6dffb8
https://rinkeby.etherscan.io/address/0x07560a88e5f17252a236f47ebc00d6eb7f6dffb8

Transaction hash contract creation:

0x82bd9640cf15b0cfb855c067f48f1bec549122e920edae231d572256b8dffabc

Infura

https://infura.io/project/975ee58ff5a144c48a603e16a15cc88f?isOnboarding=true

User: oliversimon.faust@studenti.unicam.it

Password: Camerino2020

Endpoint (rinkeby): rinkeby.infura.io/v3/975ee58ff5a144c48a603e16a15cc88f

Project id: 975ee58ff5a144c48a603e16a15cc88f

Java wrapper generation

With Web3j CLI

```
web3j solidity generate -a=<abiFile> -b=<binFile> -o=<destinationFileDir> -p=<packageName>
```

Webservice API Description:

Webservice Base Url: http://localhost:8080/CRMWebservice/rest/

Customer	**/customer**

POST /add	**Body**:
	
	`{`
	`"name": "Hansli",`
	`"familyname": "mueller",`
	`"address": "Alte Jonastrasse 24",`
	`"city": "Basel",`
	`"customerType": "A",`
	`"email": "support@bexio.com",`
	`"postcode": 4052`

	} **Returns**: Created Customer (CustomerExchange) ***Bexio Endpoint:*** `POST /contact`
GET /{id}/get	**Returns**: Customer (CustomerExchange) ***Bexio Endpoint:*** `GET /contact/4`
Offer	**/offer**

POST /create	**Body**:
	
	 `{`
	 `"contact_id": 25,`
	 `"positions": "Laptop 15 Zoll, i5",`
	 `"price": 1000,`
	 `"discount": 5,`
	 `"amount": 25`
	 `"email": "support@bexio.com",`
	
	 `}`
	
	 Returns: Created Offer (OfferExchange)
	
	 Bexio Endpoint: `POST /kb_offer`

POST /{id}/accept	**Body**: - **Returns**: Success (Boolean) ***Bexio Endpoint:*** `POST /kb_offer/1/accept`
POST /{id}/ decline	**Body**: - **Returns**: Success (Boolean)

	Bexio Endpoint: `POST /kb_offer/1/reject`

POST /{id}/send	**Body**: e-Mail address as String
	
	Returns: Success (Boolean)
	
	Bexio Endpoint:
	
	`POST /kb_offer/1/issue`
	
	`POST /kb_offer/1/send`

POST /{id}/createinvoice	**Body**: - **Returns**: Id of generated Invoice (Integer) ***Bexio Endpoint:*** `POST /kb_offer/1/invoice`
POST /offer/{id}/invoice/send	**Body**: e-Mail address as String **Returns**: Success (Boolean) ***Bexio Endpoint:*** `POST /kb_invoice/1/send`