Examensarbete, Högskolan på Åland, Utbildningsprogrammet för informationsteknik

Webb Application for customer-, account- and card register

Sebastian Alm

48:2019

Datum för godkännande: 09.12.2019
Handledare: Agneta Eriksson-Granskog

EXAMENSARBETE
Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Carl Sebastian Alm
Arbetets namn:	Webbapplikation för kund- , konto- och kortregister
Handledare:	Agneta Eriksson-Granskog
Uppdragsgivare:	Enfuce Financial Services

Abstrakt

Mitt arbete är en webbportal jag gjorde för Enfuce Financial Services. I portalen skall man kunna söka efter interna kunder och se deras information tydligt.

Man skall också kunna se kundens konton, alla kort som är under respektive konto och kontonas transaktioner. Man kan skapa nya kunder, konton, kort samt transaktioner eller editera dem.

Informationen hämtas från företagets interna API mha. en HTTP-baserad begäran där man får datan tillbaka som en respons.

Nyckelord (sökord)

Vue.js, Component, Javascript, HTML, CSS, Postman, Axios

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
48:2019	1458-1531	Engelska	33 sidor

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
10.12.2019	04.12.2019	09.12.2019

DEGREE THESIS
Åland University of Applied Sciences

Study program:	Computer Science
Author:	Carl Sebastian Alm
Title:	Web Application for customer- , account- and card register - The Enfuce Portal
Academic Supervisor:	Agneta Eriksson-Granskog
Technical Supervisor:	Enfuce Financial Services

Abstract

My project is a web portal for Enfuce Financial Services. With the portal, the user is able to search for internal customers and see their information clearly displayed.

It’s also possible to list all the customer's accounts, cards belonging to their respective account and the transactions of the accounts. The user can also create new customers, cards, accounts or transactions and edit existing entities.

The data is retrieved from the company’s internal API. With the use of an HTTP based request we get a response of the data back and choose how to use or display it.

Keywords

Vue.js, Component, Javascript, HTML, CSS, Postman, Axios

Serial number:	**ISSN:**	**Language:**	**Number of pages:**
48:2019	1458-1531	English	33 pages

Handed in:	**Date of presentation:**	**Approved on:**
10.12.2019	04.12.2019	09.12.2019

INNEHÅLLSFÖRTECKNING

1. INTRODUCTION

1.1 Purpose

This was a project made for Enfuce Financial Services where the purpose was to make use of Enfuces internal API to create a web portal where you can create, search and update entities to and from the card system. Enfuce wants to make use of the portal within the company, but they also have plans for it long term to enhance it so customers themselves have the ability to use it in the future.

The user should be able to retrieve customer information in the form of any object by searching for customer number or account number. The portal concerns information about the customer, all its accounts and cards and transactions in the accounts.

The project was carried out during the summer at the Enfuce office in Mariehamn. Enfuce is a company that offers payment and open banking services.It was the first company in the world to run a payment service platform in a public cloud.

1.2 Method

This wasn’t a one man job; it was done by me and the other intern at Enfuce. We started off using Vue.js to become familiar with the syntax and did simple simulators for the project that we built from scratch. After practicing for a while, we needed to learn how to connect via the API by code.

First, we used the Postman application that interacts with HTTP APIs. Postman offers a clear and easy GUI to build requests and responses. Postman enabled us to try out the API connection which is easier to do than in code form. Once we had established a connection on Postman we knew we could do the same on the Portal given the right syntax and values.

We made a simple design to start off with with the help of Vuetify that gave us access to a set of tabs that we could use to switch between customers. Vuetify also provided a nice spectrum of functionality and graphics for us to use. We used it to make cards where we display our information and scroll between transactions.

We didn't have a complete design template to work on, so we went with what looked simple and good enough in our eyes. The design might be improved upon later down the road. I used my knowledge in programming from my university, we didn't work too much with frontend but I had done an optional JavaScript course, which helped. I had only heard about Vue.js before, but this is when I had an opportunity to really get into it. It proved a bit different and confusing at first but it offered much quality of life compared to standard JavaScript.

We used both standard JavaScript and Vue.js, but also HTML and CSS, which I have been using from before in school, mainly in the Frontend course and Web Server Technique course. Our mentor helped us with the initial understanding of the syntax, but for most of it, we were left on our own to look up information and knowledge on the Internet and learn from there, which gave us a lot of freedom to prepare properly for the task ahead.

1.3 Limits

We were a bit limited in how we could do it. For example, it's not possible to list all customers at once and then from there click on the one you want to see the information of. Instead you have to search them one by one. We also had to make sure we retrieved the right information and not too little or too much.

Some fine-tuning and some bugs remain. It's also more of a prototype and not really "the end product". The design also might not be the best since we didn't have much to go on.

2. BACKGROUND

2.1 Enfuce

Enfuce offers payment and open banking services to banks, fintechs, financial operators, and merchants. By combining industry expertise, collaborative partnerships and compliance, they are delivering long term and scalable solutions fast and secure (Enfuce, n.d).

Enfuce was founded in 2016 by five industry experts and today the company employs over 50 driven professionals in the Nordics and has over 8 million end-users in their platform. (Enfuce, n.d).

2.2 Why the Enfuce Portal?

The Enfuce Portal is the window into the "black box" of stored data for Enfuce and hopefully for their customers as well in the future. The core idea is to search, create, view or edit entities within the card system. See Figure 1 for Enfuce's logo.

Figure 1. Enfuce Logo (Enfuce, n.d.)

3 CODING LANGUAGES/FRAMEWORK

3.1 HTML, CSS and JavaScript

HTML or Hypertext Markup Language is the standard choice for when you want designed documents to be viewable in a web browser (W3schools, n.d.).
The web browser in question can receive HTML documents locally or via a web server and in turn renders them into pages. The HTML shows you how the web page is built.
The document starts off with an <html> tag which contains typically a head and a body tag.

I was working with Vue.js though, and it has a bit of a different HTML syntax. For example, instead of the <html> tag wrapping everything, you have a <template> tag. HTML is typically used with two other languages, JavaScript and CSS, to create better functionality and design respectively (W3schools, n.d.).

CSS is an abbreviation for Cascading Style Sheets. CSS tells how the HTML elements are supposed to appear like on different media (W3schools, n.d.). CSS does a lot of work for you with little to no effort and can affect the layout of many web pages at the same time. The CSS definitions are saved in .css files or within <style> tags.

In CSS, there are so-called "selectors" that declare which part of the code a certain style applies to depending on if the tags are matching. Selectors may affect a specific type, e.g. only to all unordered lists <ul></ul> (W3schools, n.d.).

The most common HTML attributes are "class" and "id". "id" is a unique identifier, while "class" can identify multiple elements at once (HTML Dog, n.d.). CSS classes and IDs are case sensitive, i.e. a program differentiates between uppercase letters and lowercase letters. In Figure 2 is a code example of how CSS changes the color of HTML text:

```
<h1 style="color: orange;"> myText </h1>

<style>
h1 {
    color: orange;
}
</style>
```

Figure 2. CSS code example

3.2 JavaScript

JavaScript is code that runs as the web page loads into memory. Once you have written JavaScript code, you have essentially created a so-called script. Unlike Java, which requires compilation to run its code, JavaScript can be run as simple text. JavaScript can be run in browsers, on servers or any device that has a program with a JavaScript engine installed.

A few examples of what JavaScript can do in a browser (Javascript, 2019):

- Change existing HTML or add new
- Modify CSS
- React to different actions such as key input or mouse clicks.
- Have a dialogue with the user
- Store data locally

In the portal we were working with Vue.js, which I will explain more about further in the text. Vue.js utilizes HTML, JavaScript and CSS. We used HTML combined with "Vue-specific" tags to get the necessary information on the screen. We also utilized CSS to change layout, color, font and more. Vue.js is very flexible, so we were able to use "basic" JavaScript where we felt it easier to implement and understand.

3.3 Vue.js

Vue.js or "Vue" is an open source (source code that is accessible to anyone) framework for JavaScript, used for convenient building of user interfaces and applications on the web page. Vue is designed to give the user the possibility to integrate different libraries together with the core library, which is only for the view level alone (Vue.js, n.d.). See Figure 3 for the Vue.js logo.

Figure 3. The Vue.js logo (Vue.js, n.d.)

3.3.1 History

After developer Evan You had worked for Google using Angular.js, he wanted to create something more lightweight, combined with the parts he liked about Angular. He began the project in July 2013 and Vue released the next year in February 2014 (Between the Wires, 2014).

3.3.2 DOM

A virtual DOM stands for a Virtual Document Object Model. The "virtual" version of a UI is stored in memory and combined with the original DOM by a library like ReactDOM. One can think of a Virtual DOM as a duplicate DOM that can be changed without using the original DOM's APIs. A method asks React in which state you want your UI to be at, and it guarantees that the DOM is appropriate for that state (React, n.d).

3.3.3 Templates

Vue.js is a framework that uses a template syntax that is based on HTML. It lets the DOM be bound to the Vue instance's data. The templates are parsed by HTML parsers (analyzers of a string of symbols) as well as specification compliant browsers.

The templates are compiled by Vue into Virtual DOM rendering functions. With the help of the reactivity system, Vue is also able to re-render the minimal required number of components to re-render and figure out the minimal DOM manipulations necessary when it's app state changes (Vue.js, n.d).

3.3.4 Reactivity

The Vue framework offers a reactivity template that uses simple JavaScript objects and simplifies re-rendering. Each module monitors its sensitive dependencies throughout the rendering process, so the system knows exactly when to render and what components to render again. There are a few advantages and disadvantages with using Vue.js. That can be read in the Pros and Cons with Vue.js (Medium, 2018, Mar 16).

Pros with Vue.js

- Improved HTML - If a programmer is familiar with basic HTML he can get into the Vue HTML template easily and also use the basic HTML if needed.

- Detailed Documentation - Vue.js has a fully detailed documentation which makes it easy for developers who know the basic JavaScript and HTML to learn, as was the case for me.
- Adaptability - It’s similar to other frameworks such as Angular and switching from one to the other wouldn’t take too long.
- Small Size - It doesn’t take a lot of space, so using Vue.js ensures good performance.

Cons with Vue.js

- **Lack of resources** - Vue.js is still quite new, so finding other peoples’ knowledge or help with solutions while using Vue might be a bit tricky. I noticed this, as I had to mostly rely on tutorials or official documentation. Yes, there’s very detailed documentation out there, but it can be hard to find specific information about it.

3.3.4 Components

Components are different parts of your Vue program. One component is a separate reusable Vue instance with its own name. It’s easy to think of them as the code you would normally organize and split up into different parts (different components in this instance) where each component does it’s own thing and the other components can depend on each other when needed. You could, for example, have one component that handles navigation links, one component that handles buttons and one component that handles the text (Vue.js, n.d.).

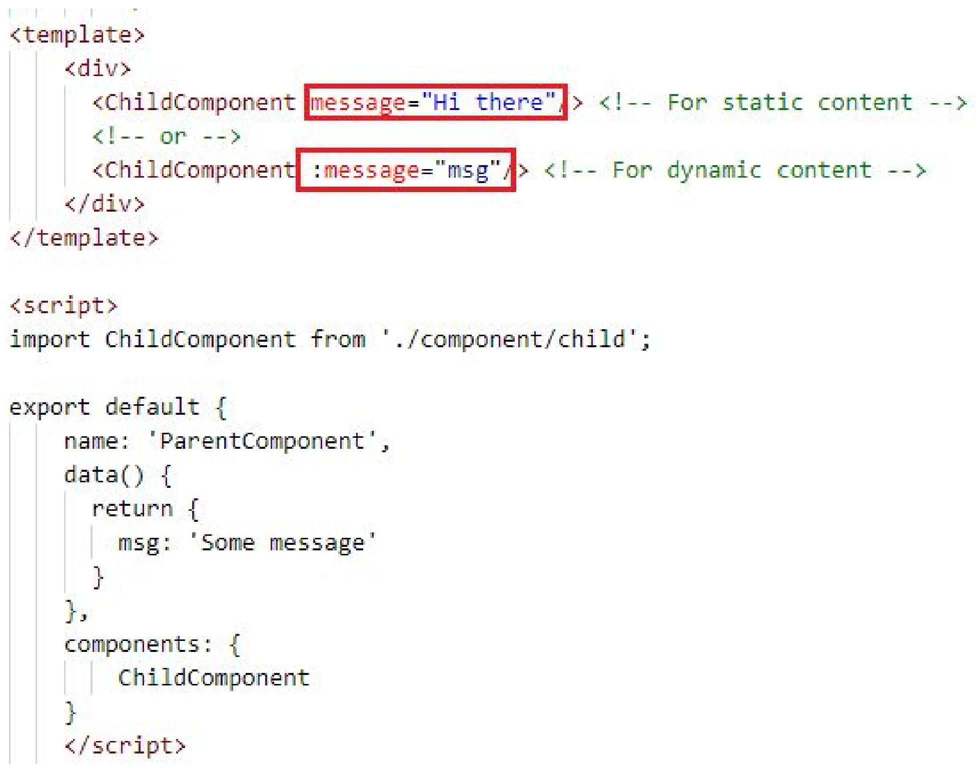
Your main component is called a Parent component and its other components are Child components. Vue.js allows communication between these components in the following ways

- Parent to child
- Child to parent
- Communication between all components

It does this via emits or event buses.

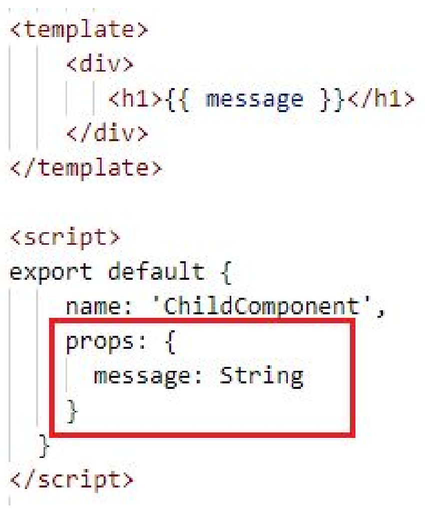
3.3.5 Parent to Child

In a “parent to child” communication, the parent component sends its data to a child component. This the parent does by using “props”, which simply is an array of one or many data elements. When the prop is changed in the parent component, the change update is sent to the child component, where it re-renders the information. It’s not optimal to try and change a prop from a child to parent component. We used this in our project for example to activate a block of code that shows the account Information under a customer in a child component by clicking on a button that exists in the parent component (Medium, 2019). Here in Figure 4, is an example of how a “parent to child” connection is done by code



```
<template>
    <div>
      <ChildComponent message="Hi there"/> <!-- For static content -->
      <!-- or -->
      <ChildComponent :message="msg"/> <!-- For dynamic content -->
    </div>
</template>

<script>
import ChildComponent from './component/child';

export default {
    name: 'ParentComponent',
    data() {
      return {
        msg: 'Some message'
      }
    },
    components: {
        ChildComponent
    }
    </script>
```

Figure 4. An argument ***message*** *is added where the child component is made (Medium, 2019)*



```
<template>
    <div>
      <h1>{{ message }}</h1>
    </div>
</template>

<script>
export default {
    name: 'ChildComponent',
    props: {
      message: String
    }
  }
</script>
```

Figure 5. This is where the input is declared in the child component (Medium, 2019)

3.3.6 Child to Parent

When communicating from a child component to a parent component, you utilize Events, which the parent component receives by the “v-on” or “on-click” event handlers that are declared in the template, exemplified here in Figure 6:

```
myButton: function() {
 this.$emit('buttonActivated')
 }
```

Figure 6. Emit being sent when triggering this function (Flavioscopes, 2018)

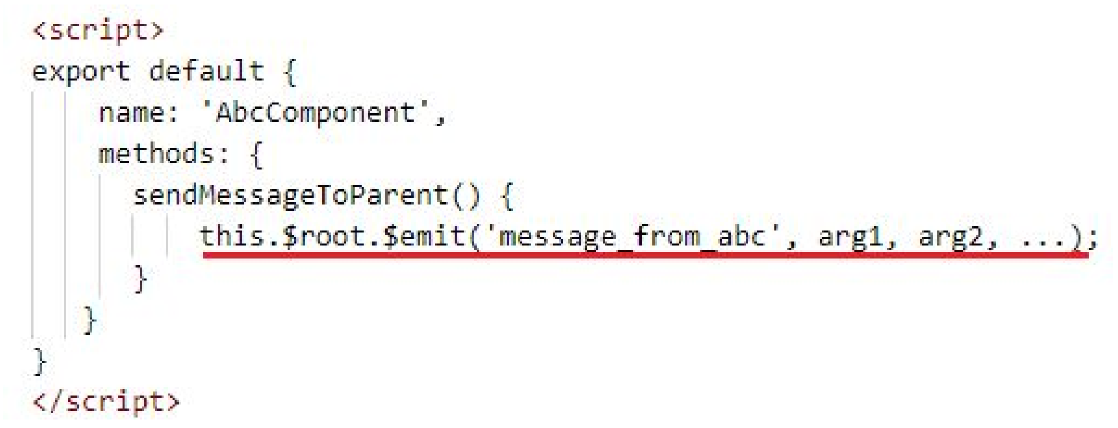
In this example, the event is being triggered by the click of a button in the child component, which activates a function in the parent component (Flaviocopes, 2018) . This will then activate the parent function and run whatever code is inside of it as shown in Figure 7:

```
myParentFunction: function() {
//do this...
}
```

Figure 7. The parent function that is called when the emit is triggered (Flavioscopes, 2018)

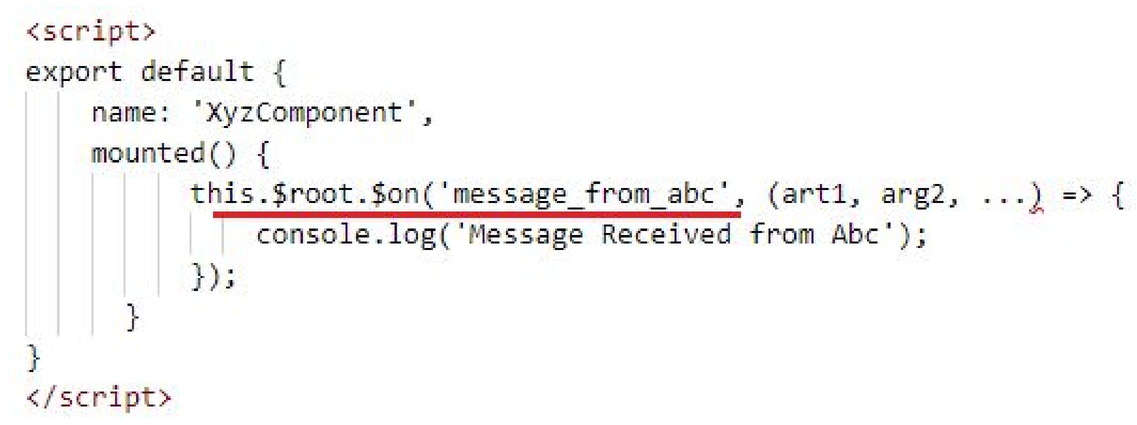
3.3.7 Event Bus

Unlike “Child to Parent” or “Parent to Child” components, which can only send information in a “one-way-street”. Event Buses can be used between any kind of components as shown in Figure 8 and 9.



```
<script>
export default {
    name: 'AbcComponent',
    methods: {
      sendMessageToParent() {
        this.$root.$emit('message_from_abc', arg1, arg2, ...);
      }
    }
}
</script>
```

Figure 8. Event bus step 1 (Medium, 2019)



```
<script>
export default {
    name: 'XyzComponent',
    mounted() {
          this.$root.$on('message_from_abc', (art1, arg2, ...) => {
            console.log('Message Received from Abc');
          });
    }
}
</script>
```

Figure 9. Event bus step 2 (Medium, 2019)

In the example above, first you emit an event from the “AbcComponent” that the “XyzComponent” is listening for. This is done by adding the “mounted” lifecycle hook that is activated when the event is called from any other component (Medium, 2019). In our project we didn’t utilize these much, so I don’t have too much experience with them.

We also utilized Events in our project to a great extent, so we could have most of our functions in our parent component, which was easiest, because the biggest amount of data was there, and the relevant child components activated the relevant functions in the parent component. For example, a button in our Account component triggers an event that lets the parent component show the account data under a customer.

Vue.js was the main framework we used throughout the project. Initially we got some time to learn and adapt to the framework and see how different it can be compared to working with basic JavaScript. Thanks to a lot of tutorials and completed mini tasks, we were able to learn it to an extent, which was required for the Enfuce Portal to be finished. Previous knowledge from school courses also played a big part in helping to understand Vue.js.

3.4 NPM

NPM stands for Node Package Manager and can be described as follows: It is mainly an online database for publishing open source Node.js projects. NPM is also a command line interface used to communicate with the registry that facilitates version control, package installment and dependency management.

NPM contains many programs and libraries and more are constantly being added as well. By visiting the website **http://search.npmjs.org/** ,it's possible to search the libraries and install them once the desired one has been found. This you do by running the command **npm install -library name-** in the terminal. The specified library will be added into a "node-modules" folder where it can then be used from like they were built in the application. It's also possible to globally install the application on your device, one only needs to add -g after the chosen library in the command (npm Documentation, n.d.).

For our project, we used npm to set up the Vue instance and to install various libraries for our use. For example, there was one library that enabled us to use a simple pop up window.

3.5 Node.js

Node.js is a runtime environment that's both open source and cross-platform for creating server-side and network applications. Node.js applications are programmed in JavaScript and can be run on Windows, Linux and OS X. Node.js streamlines web application development by using a vast collection of different JavaScript modules (Tutorialspoint, n.d.).

Node.js became a runtime environment when the developers of JavaScript upgraded it to something that's possible to run on your PC as its own application rather than only relying on a browser. This gives JavaScript more to do than just activating a website. Node.js is run on the V8 JavaScript runtime engine which takes JavaScript code and accelerates it into quicker machine code, meaning code that the computer runs without needing to parse it first. (Patel 2018) We installed Node.js early on so we could use NPM properly.

3.6 Axios HTTP requests

To communicate with servers via a website, it’s common to use the HTTP protocol where you can make HTTP requests and retrieve a response with data. Axios is an HTTP API that is promise based, which means it’s in a pending, fulfilled or rejected state
(MDN web docs, 2019).

Axios is installed by using NPM with the command: “npm install axios”. Making an HTTP request with Axios isn’t too complicated. One just has to know which objects the server has, so you know what you’re requesting in the code. Here in figure 10 is an example of how to make an HTTP POST request with Axios:

```
this.$axios.$post('myWebserver.com’, {
     {
               email: “myEmail@hotmail.com”,
               phoneNumber: “+358443005”

            })
            .then(response => {
              console.log("Customer Created");
```

Figure 10. Axios POST example

We chose to use the POST request, which means you add something new in the server. Then we can fill in any fields we want to declare when creating the new entity, and then we get a response back with the newly added data, and we can get a confirmation that it worked by logging in the console. If it doesn’t work properly, you get an error with status code 500 or 400, depending on what the error is.

3.6.1 Requests

Aside from POST requests, there are also GET requests, which you use to retrieve the existing data, and then do what you want with it. We used GET to store the data in a variable and then print it out on our portal. We could also use a GET to compare other data with each other, for example, we could show a customers account by comparing their relevant IDs to each other, which we retrieved with a GET command.

The third request we used is PATCH, which takes existing data and changes it, either by posting new fields that are null or changing an Email address. The difference between PATCH and POST in our project is that we were only using POST when creating new objects, and PATCH when we wanted to change existing ones.

There's also a DELETE request, which as the word suggests, deletes an existing object. We, however, didn't use this request at all because the company's API doesn't offer that functionality, so a DELETE request simply wouldn't work in this case (Zetcode, 2019)

We used Axios requests a lot in the Portal to POST (Create), GET (Fetch) or PATCH (Update) our Customer, Account and Card objects. It was the most fun to figure out how to handle the **responses** and display the data in a readable manner.

3.6.2 Handling Axios responses

When a Axios request has been made, we get a response, which we can work with or make something happen once a response has been received. This is done within the response bracket. In the following example, I'm doing a simple log in the console, but you can also use the response itself as data you pass to a variable and print it out on the screen. Figure 11 shows how we wrote the code in our program to check that the request was successful:

```
methods: {
//Gets the customer and all information connected to
it
          async getCustomer(id){
            await this.$axios.$get('myWebserver.com'+
id +'?auditUser=getCustomer', {
            })
            .then(response => {
              console.log("Response successful!");
```

Figure 11. Handling an Axios response

3.7 Postman

Postman is an app which helps the user test their APIs. It’s able to commit HTTP requests, including the ones we spoke about in previous sections. Postman also transforms your API into different programming language codes like JSON or Python. Here I’ve listed some advantages and disadvantages with using Postman (Geeksforgeeks, n.d).

Pros with Postman

- Simple API - easy to understand
- Shows status codes on requests, so you have an idea of the error
- Possible to import existing APIs

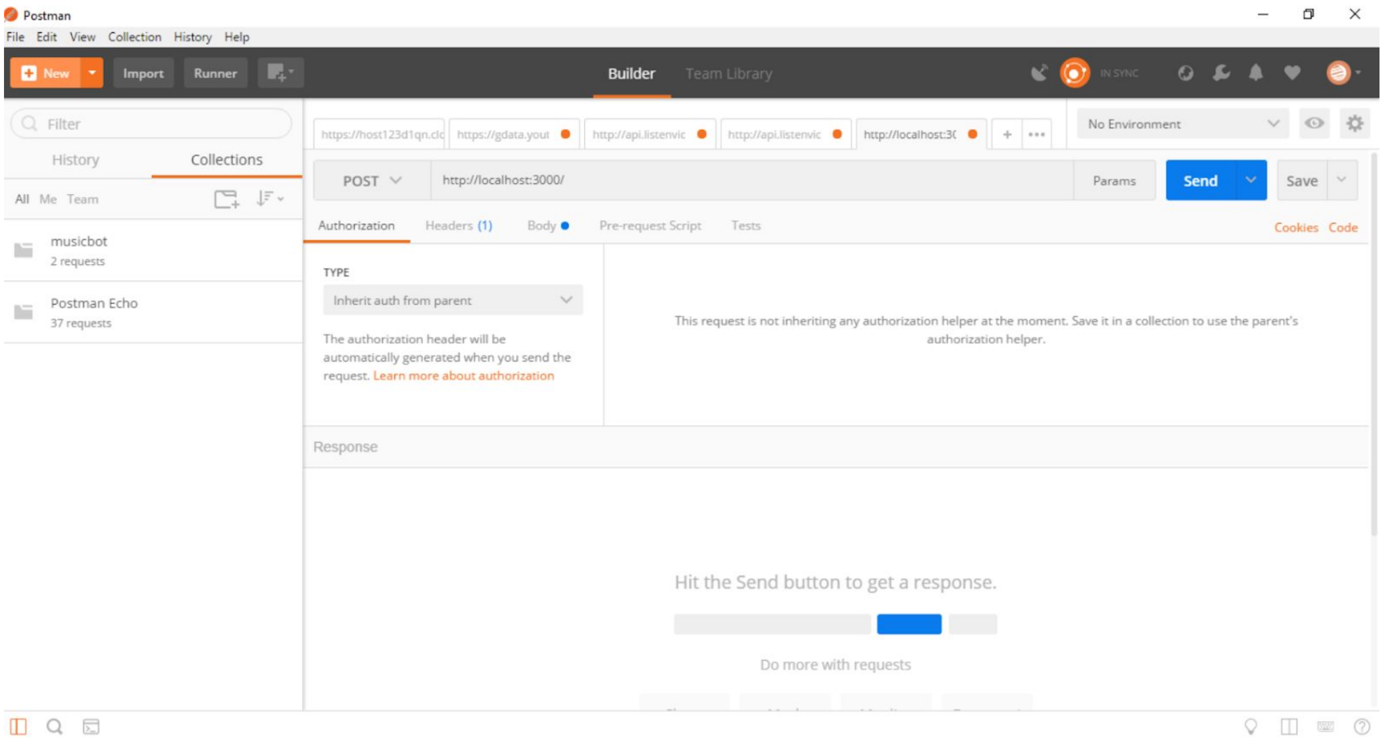
Cons With Postman

- A lot of options for a beginner
- No guarantee for a Postman-made API to work in browser as well

We used Postman in our project because it offers a very user-friendly interface and a straightforward way to test out API connections. We could import the existing requests that Enfuce were already using and start working from there.

Basically we were able to test the connection, the requests and the responses through Postman. If the request worked, we knew how the request would look like in our code. If we got an error 400 in our web browser instead, but a passed request in Postman, we knew that the error was in our coding. And if we couldn’t make a request in both Postman and our portal, then something was wrong with the connection or documentation, which we then needed help with or wait for a fix before we could do further testing.

As mentioned, it’s very simple to use, as long as you know what kind of request you want to make, the necessary objects and how to write the request properly. The Postman interface is displayed in Figure 12.



POST http://localhost:3000/

TYPE: Inherit auth from parent

The authorization header will be automatically generated when you send the request. Learn more about authorization

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Hit the Send button to get a response.

Figure 12. Postman interface (Geeksforgeeks, n.d.)

4. GUI (GRAPHICAL USER INTERFACE)

4.1 Design

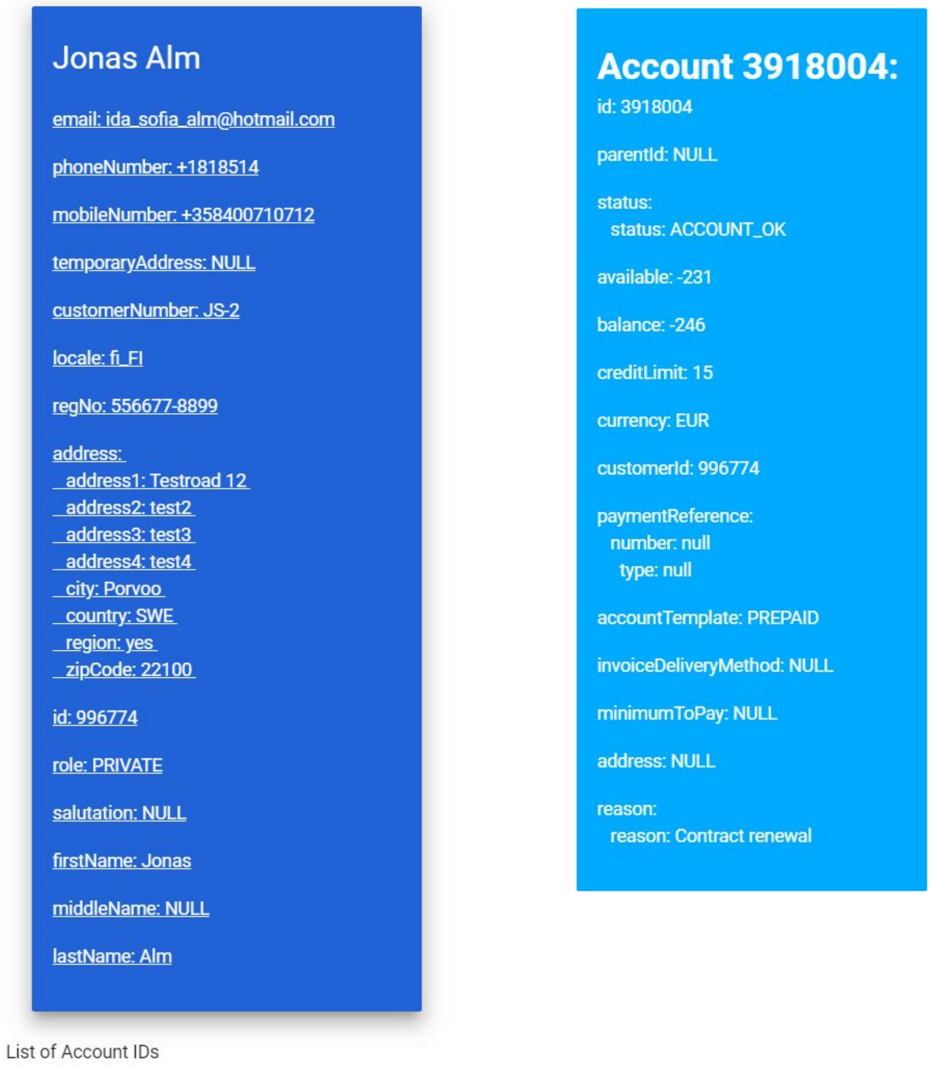
It wasn't enough to just print out the API information on the browser. We had to figure out how to display it in a conveniently readable way. We didn't have a clear design plan to follow, so we went with what felt good from our perspective, with the help of CSS and Vuetify, a component framework developed for Vue alone.

As mentioned, we didn't use any predefined template. We were told to simply display the information of a customer or account by searching for the corresponding ID. The idea we went with was to begin with displaying basic customer information. Then we wanted to list all the accounts belonging to that customer and after that list cards and transactions under their respective accounts.

4.2 Vuetify

Vuetify is a Vue.js framework that supports most current browsers. Vuetify gives the user access to simple, reusable components which are easy to implement via the HTML template (Medium, 2018, Sep 19).

We used Vuetify for minor but useful things like adding information on a clear background, called "The v-card component" (Vuetify, n.d). We could then place every API object and its information inside these cards, so we made one card for each category (customer, account etc.). It's easy to implement: You implement the **v-card** tag in the HTML template and everything that goes inside that tag ends up in the actual "card" like in Figure 13



```
Jonas Alm

email: ida_sofia_alm@hotmail.com

phoneNumber: +1818514

mobileNumber: +358400710712

temporaryAddress: NULL

customerNumber: JS-2

locale: fi_FI

regNo: 556677-8899

address:
  address1: Testroad 12
  address2: test2
  address3: test3
  address4: test4
  city: Porvoo
  country: SWE
  region: yes
  zipCode: 22100

id: 996774

role: PRIVATE

salutation: NULL

firstName: Jonas

middleName: NULL

lastName: Alm
```

```
Account 3918004:
id: 3918004

parentId: NULL

status:
  status: ACCOUNT_OK

available: -231

balance: -246

creditLimit: 15

currency: EUR

customerId: 996774

paymentReference:
  number: null
    type: null

accountTemplate: PREPAID

invoiceDeliveryMethod: NULL

minimumToPay: NULL

address: NULL

reason:
  reason: Contract renewal
```

List of Account IDs

Figure 13. Customer and account information displayed on v-card components

We also used the **v-button** component to easily implement good looking buttons as shown in Figure 14.

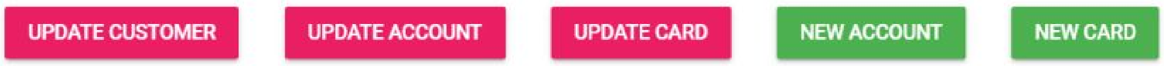
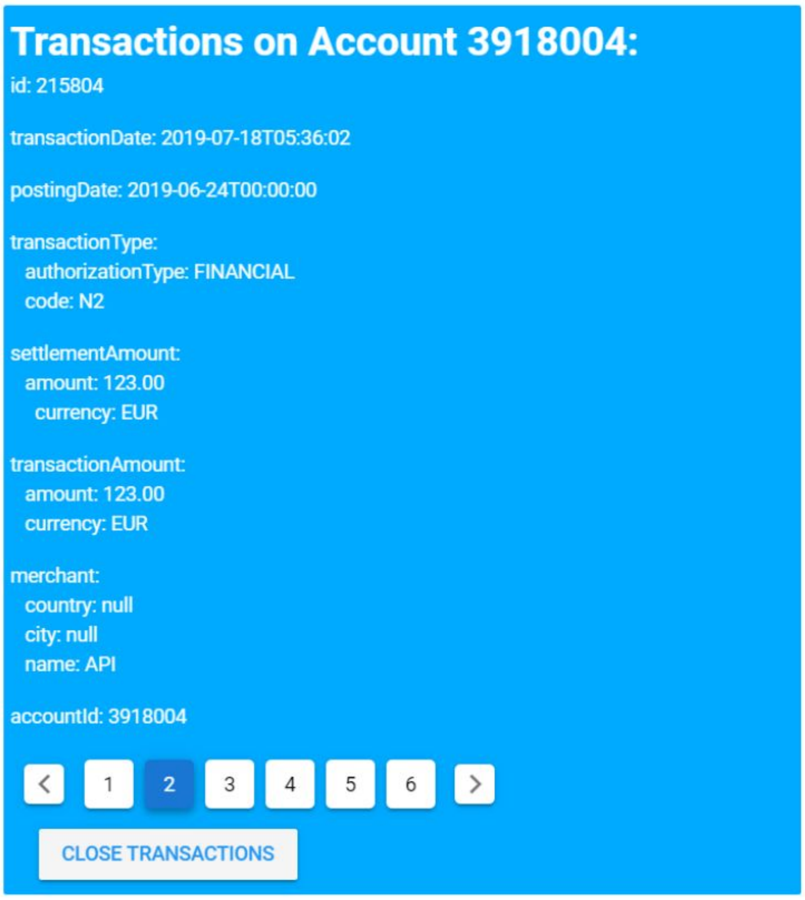


Figure 14. Various buttons for updating entities or creating new ones

We were able to easily insert paginations into the interface with the use of the “v-pagination” component. The component is commonly meant to split a big amount of data so that it becomes easier to read. The v-model directive is used to keep the current page within the pagination.

The pagination shows the number of pages based on the value of the length property. You can go to the next or previous page by clicking the arrows at both ends of the pagination (Vuetify, n.d).

We used the pagination component to display all transactions under one account, since the transaction data for one account can be quite large in some cases. Every transaction object is in an array, so we bound the length property to the amount of elements in the data. Then by clicking the next button, you go to the next position in the array, and then that transaction's data would appear within the v-card like this in Figure 15:



```
Transactions on Account 3918004:
id: 215804

transactionDate: 2019-07-18T05:36:02

postingDate: 2019-06-24T00:00:00

transactionType:
  authorizationType: FINANCIAL
  code: N2

settlementAmount:
  amount: 123.00
   currency: EUR

transactionAmount:
  amount: 123.00
  currency: EUR

merchant:
  country: null
  city: null
  name: API

accountId: 3918004
```

Figure 15. The transaction v-card with pagination at the bottom.

A Vuetify component we used early on and that was a core part of the interface was “v-tabs”. By implementing these, we were able to add, delete and scroll through tabs, which I mentioned earlier. A new tab will open up a new page where you search for a customer. Switching the tabs didn’t reset the data. By clicking the “+” next to the last tab, you push a new tab item to the array, this is displayed in Figure 16.

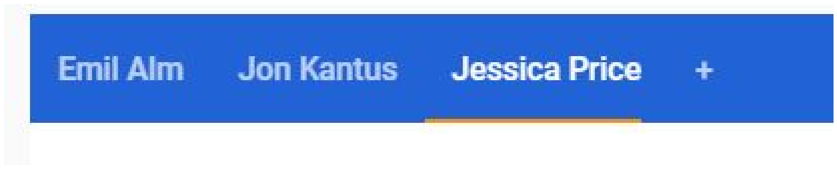


Figure 16. Tabs for switching through customers or creating new ones to search for

At the bottom of every page, there’s a “CLOSE” button that closes the page and splices the corresponding item from the tab array. See Figure 17 for the button.

Figure 17. Tabs for switching through customers or creating new

We used CSS to change things inside the Vuetify components themselves like font, header and text color. More figures that displays the portal’s layout are found in Appendices 1-3 and a creation form is displayed in Appendix 4.

5. RESULTS

5.1 Final Analysis

The project itself was a fair challenge. Enfuce gave us enough time to practice Vue.js and get familiar with the framework to know how to proceed. Each week, we focused on one task and we made steady progress. Some hiccups occured due to the configuration being faulty or us struggling to establish a proper API connection several times, which lead to some time loss. The other struggle was the design. At least personally I wasn't sure how to make it all come together. We just went with our intuition in creating something accessible. It was also a nice learning experience for developing something for a client. I tried to view the program from the user's perspective and I came up with new ways to improve the program because of it. All in all, it was fun to do, and this project has made me more confident as a developer.

5.2 Future Work

I wouldn't mind to work some more on the project, as we were in the middle of cleaning up the code. I might get another chance to take a look at it in the future again.
The project is planned to be continued. It was meant to be used internally at Enfuce at first and later on develop it further, so their customers can access it. What we made was, as said, a prototype of sorts.

REFERENCES

Enfuce. (n.d.) Who we are. Retrieved from: https://enfuce.com/about/who-are-we/

Enfuce (n.d.). Enfuce logo [Picture] Retrieved from: https://enfuce.com

Javascript (November 2019). An introduction to Javascript. Retrieved from:
https://javascript.info/intro

HTML Dog (n.d.). Class and ID Selectors. Retrieved from:
https://www.htmldog.com/guides/css/intermediate/classid/

W3schools. (n.d.) What is CSS?. Retrieved from:
https://www.w3schools.com/css/css_intro.asp

Medium (2018, Mar 16). React vs Angular vs Vue.js — What to choose in 2019?
Retrieved from:
https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

Vue.js. (n.d.) Vue logo [Picture]. Retrieved from: https://vuejs.org

Between the Wires. (2016, Nov 3) Evan You. Retrieved from:
https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/

W3schools. (n.d.) CSS Introduction. Retrieved from:
https://www.w3schools.com/css/css_intro.asp

W3schools. (n.d.) CSS Selectors. Retrieved from:
https://www.w3schools.com/css/css_selectors.asp

Geeksforgeeks. (n.d.) Introduction to Postman forAPI Development Retrieved from:
https://www.geeksforgeeks.org/introduction-postman-api-development/

Vue.js (n.d.) Introduction. Retrieved from https://vuejs.org/v2/guide/

Vue.js (n.d.). Template Syntax. Retrieved from: https://vuejs.org/v2/guide/syntax.html

React. (n.d.) Virtual DOM and Internals. Retrieved from:
https://reactjs.org/docs/faq-internals.html

Vue.js (n.d.). Component Basics. Retrieved from https://vuejs.org/v2/guide/components.html

Medium. (2019, Apr 8) Component Communication in Vue.js. Retrieved from:
https://medium.com/js-dojo/component-communication-in-vue-js-ca8b591d7efa

Flaviocopes. (2018, Jun 20). Retrieved from:
https://flaviocopes.com/vue-components-communication/

Medium. (2019, Apr 8) Code Examples [Copied Code] Component Communication
https://medium.com/js-dojo/component-communication-in-vue-js-ca8b591d7efa

npm Documentation (n.d.) Npm Javascript Package Manager. Retrieved from:
https://docs.npmjs.com/cli/npm.html

TutorialsPoint. (n.d.) Node.js - Introduction. Retrieved from:
https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

freeCodeCamp. (2018) What exactly is Node.js? Retrieved from:
https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/

MDN web docs. (2019, Aug 10) Promise. Retrieved from:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Medium. (2018, Sep 19) Choosing the Right Front-End Framework for Your Vue App.
Retrieved from:
https://medium.com/the-web-tub/choosing-the-right-front-end-framework-for-your-vue-app-4448bac12ce7

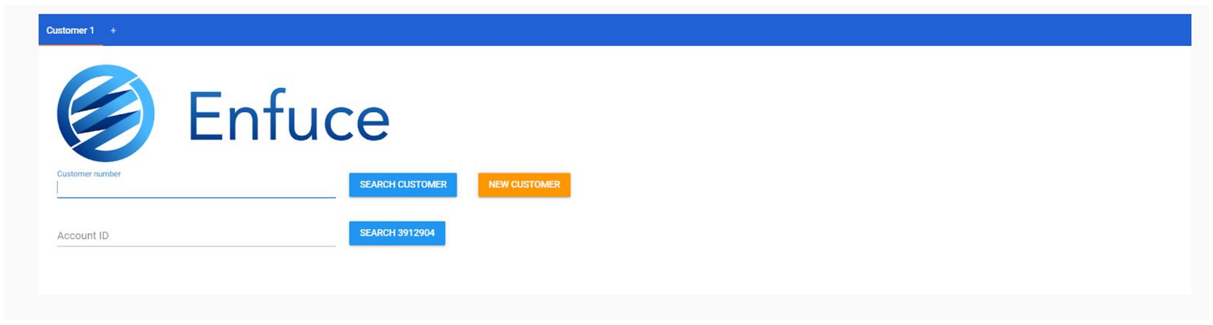
Vuetify. (n.d.) Cards. Retrieved from: https://vuetifyjs.com/en/components/cards

Vuetify. (n.d.) Paginations. Retrieved from: https://vuetifyjs.com/en/components/paginations

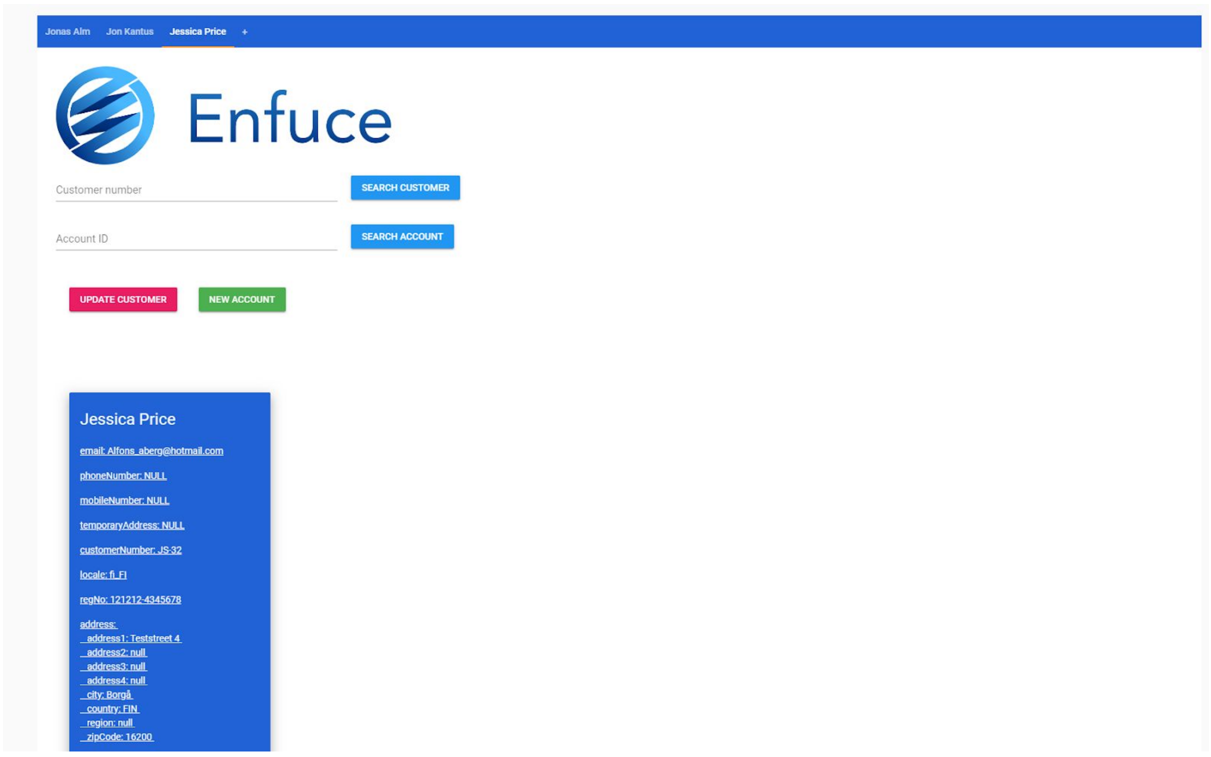
Zetcode. (2019, Feb 4) Axios Tutorial. Retrieved from: http://zetcode.com/javascript/axios/

APPENDICES

Appendix 1: Initial view

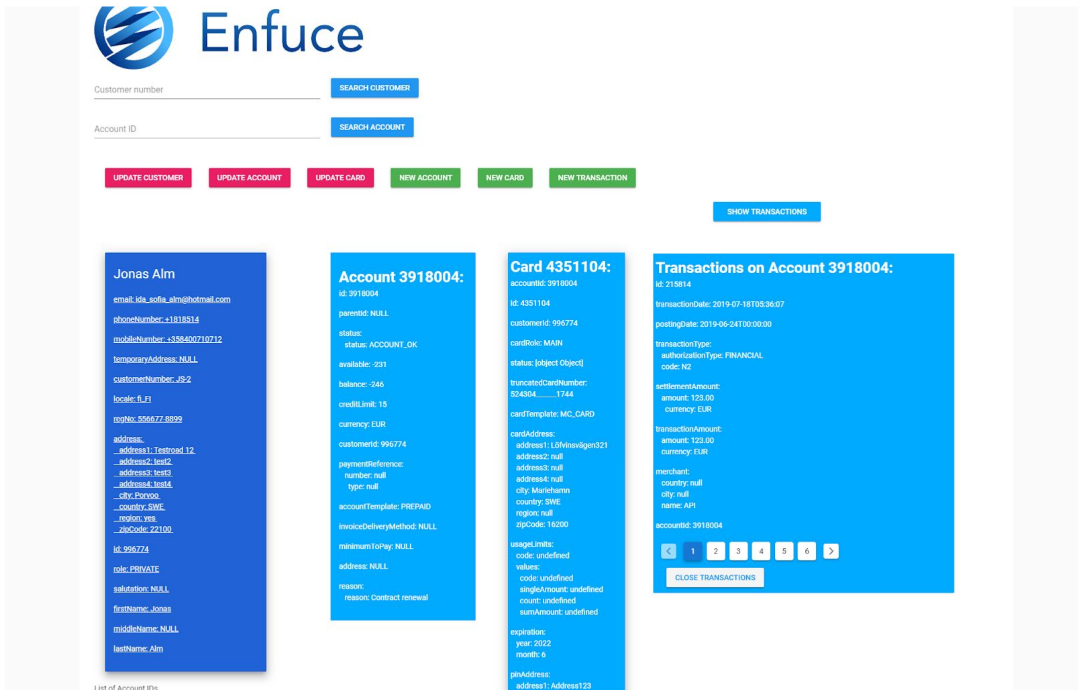


Appendix 2: After customer search



```
Jessica Price

email: Alfons_aberg@hotmail.com

phoneNumber: NULL

mobileNumber: NULL

temporaryAddress: NULL

customerNumber: JS-32

locale: fi_FI

regNo: 121212-4345678

address:
  address1: Teststreet 4
  address2: null
  address3: null
  address4: null
  city: Borgå
  country: FIN
  region: null
  zipCode: 16200
```

Appendix 3: Additional information view



Jonas Alm

email: ida_sofia_alm@hotmail.com

phoneNumber: +1818514

mobileNumber: +358400710712

temporaryAddress: NULL

customerNumber: JS-2

locale: fi_FI

regNo: 556677-8899

```
address:
  address1: Testroad 12
  address2: test2
  address3: test3
  address4: test4
  city: Porvoo
  country: SWE
  region: yes
  zipCode: 22100
```

id: 996774

role: PRIVATE

salutation: NULL

firstName: Jonas

middleName: NULL

lastName: Alm

List of Account IDs

Account 3918004:

```
id: 3918004

parentId: NULL

status:
  status: ACCOUNT_OK

available: -231

balance: -246

creditLimit: 15

currency: EUR

customerId: 996774

paymentReference:
  number: null
   type: null

accountTemplate: PREPAID

invoiceDeliveryMethod: NULL

minimumToPay: NULL

address: NULL

reason:
  reason: Contract renewal
```

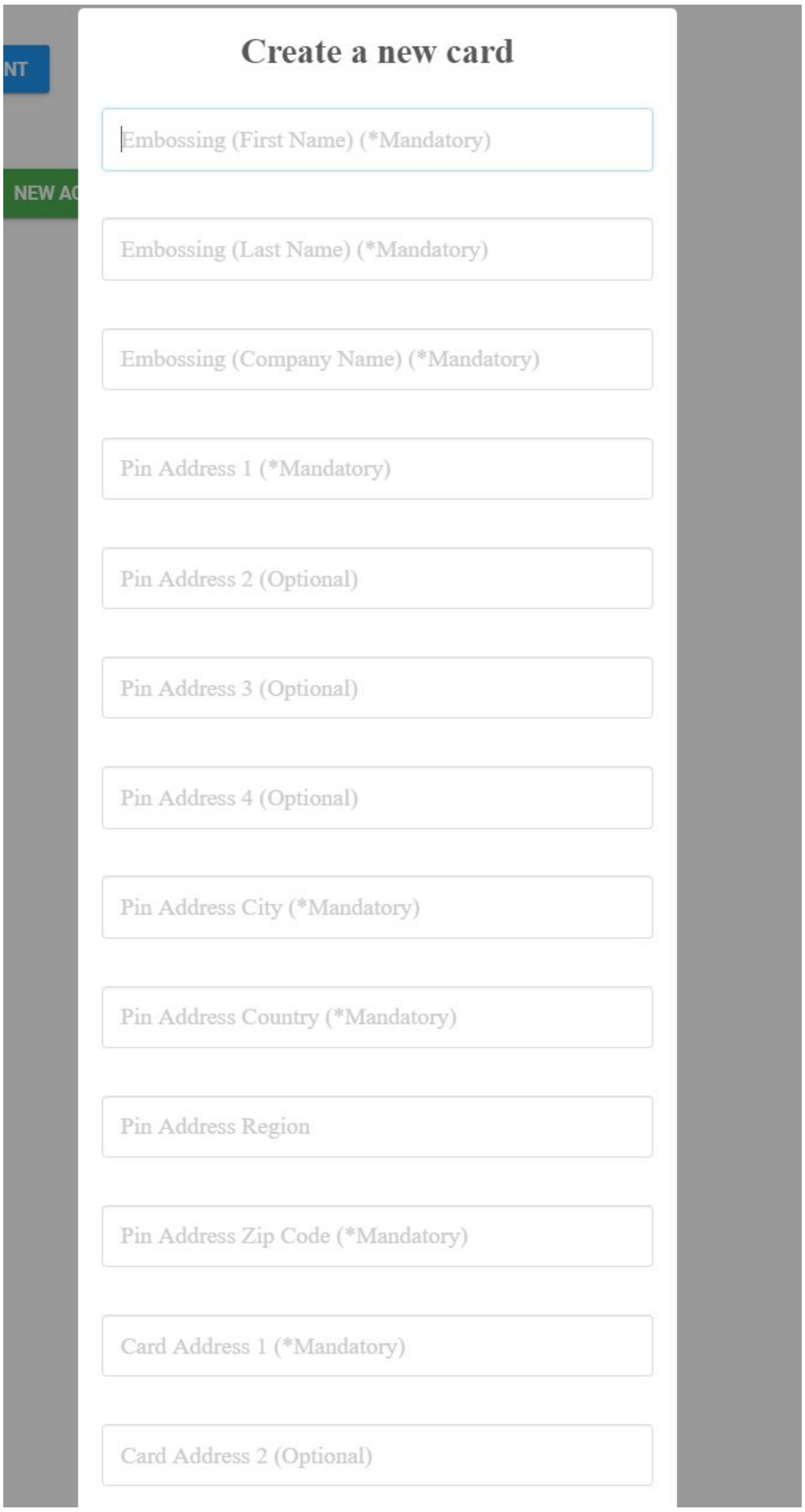
Card 4351104:

```
accountId: 3918004

id: 4351104

customerId: 996774

cardRole: MAIN

status: [object Object]

truncatedCardNumber:
524304______1744

cardTemplate: MC_CARD

cardAddress:
  address1: Löfvinsvägen321
  address2: null
  address3: null
  address4: null
  city: Mariehamn
  country: SWE
  region: null
  zipCode: 16200

usageLimits:
  code: undefined
  values:
    code: undefined
    singleAmount: undefined
    count: undefined
    sumAmount: undefined

expiration:
  year: 2022
  month: 6

pinAddress:
  address1: Address123
```

Transactions on Account 3918004:

```
id: 215814

transactionDate: 2019-07-18T05:36:07

postingDate: 2019-06-24T00:00:00

transactionType:
  authorizationType: FINANCIAL
  code: N2

settlementAmount:
  amount: 123.00
   currency: EUR

transactionAmount:
  amount: 123.00
  currency: EUR

merchant:
  country: null
  city: null
  name: API

accountId: 3918004
```

Appendix 4: Creation form with fields



Create a new card

Embossing (First Name) (*Mandatory)

Embossing (Last Name) (*Mandatory)

Embossing (Company Name) (*Mandatory)

Pin Address 1 (*Mandatory)

Pin Address 2 (Optional)

Pin Address 3 (Optional)

Pin Address 4 (Optional)

Pin Address City (*Mandatory)

Pin Address Country (*Mandatory)

Pin Address Region

Pin Address Zip Code (*Mandatory)

Card Address 1 (*Mandatory)

Card Address 2 (Optional)