

Soneso GmbH

ICO Payment

Lumenshine- „ICO Payment for arbitrary coins” – V. 1.3

Table of Contents

1.	Table of Changes	3
2.	Introduction	4
2.1	Purpose	4
2.2	Intended Audience.....	4
2.3	Scope	4
3.	General idea	4
4.	Placing a new order.....	5
4.1	Overview.....	5
5.	Viewing existing orders	16
5.1	Overview.....	16
6.	Processing orders at login	17
7.	Server Interfaces	19
7.1	Details for ICO Phase	19
7.2	Calculated payment sum	19
7.3	Create Order	19
7.4	Order details	19
7.5	Get payment transaction	21
7.6	Execute payment transaction	21
7.7	Get order list.....	21
8.	Images.....	21

1. Table of Changes

Version	Description	Changed by	Changed on
1.0	First version	Christian Rogobete	16.07.2018
1.1	Update of general process	Christian Rogobete	17.07.2018
1.2	Refactoring for new logic	Christian Rogobete	14.09.2018
1.3	Refactoring and adding new screens	Christian Rogobete	17.09.2018

2. Introduction

2.1 Purpose

This specification describes the payment process in an ICO by using the Lumenshine Tool Suite and Wallet.

2.2 Intended Audience

The intended audience for this document are Soneso & partner companies project managers, designers, server and client developers, product testers and documentation writers.

2.3 Scope

The application is used as a wallet based on the stellar network. In addition, ICO specific features can be displayed, so that the Lumenshine wallet can optionally be used for the sale of specific tokens. It allows users to register, create stellar accounts and use it for transactions. The wallet is supposed to offer the highest possible usability. For example, the user should not have to enter stellar account secrets for transactions, the wallet should persist over multiple devices, it should support the different tokens provided in the stellar network and offer other features such as the possibility to create shared accounts.

This document describes the functionality and the screens of the payment process including wallet and admin frontend as well as definition of the backend interfaces.

3. General idea

This chapter describes the general idea behind the payment process. The single use cases with the corresponding functionality are described in detail in further chapters.

The user must be logged in to be able to order ICO coins/assets/tokens. To do so she places an ICO coin order. When placing an order the user can select the currency she would like to use (such as BTC, ETH, XLM or FIAT) for purchasing the arbitrary tokens. As soon as the user placed an order, the deposit address or bank account is displayed to her.

The user must send the exact amount of currency displayed in the order to the deposit address or bank account. If the amount does not match, then the order is not filled and the user is informed. In this case, the received (crypto-) amount will automatically be refunded to the user (less any transaction costs). Each ICO Phase has a limited number of coins. After these are consumed, no further order can be filled. That's why the rule "First pay first serve" applies. If a user sends a payment that can no longer be accepted, it will be refunded less any transaction costs.

As soon as the payment has been received on the deposit address, the payment can be initiated. The user is informed and must enter his password, so that the wallet and the trustline to the arbitrary coin can be created. Since the wallet needs to be funded, the wallet and trustline is only created after receiving the user's payment. If the user is logged out when informed, the wallet and trustline is created on next login.

As soon as the wallet and the trustline have been created, the ordered coins/assets will be transferred to the user and the order will be marked as filled. But even if the trustline exists, we still need the user interaction because sending the ordered coins implies the users signature. This is because we use the users stellar account as a channel to send the coins from the distribution account. For this, the client needs to request the payment transaction for signing after making sure that the account is created and the trustline exists. This means that the users password must be requested anyway, even if the account and trustline already exists (e.g. from previous ICO Phases). If this process is executed at login, there is no need for the client to request the password again, because it already has it from the login.

4. Placing a new order

4.1 Overview

To place a new order, the user presses the "Order coins" button of the ICO card displayed in his client.

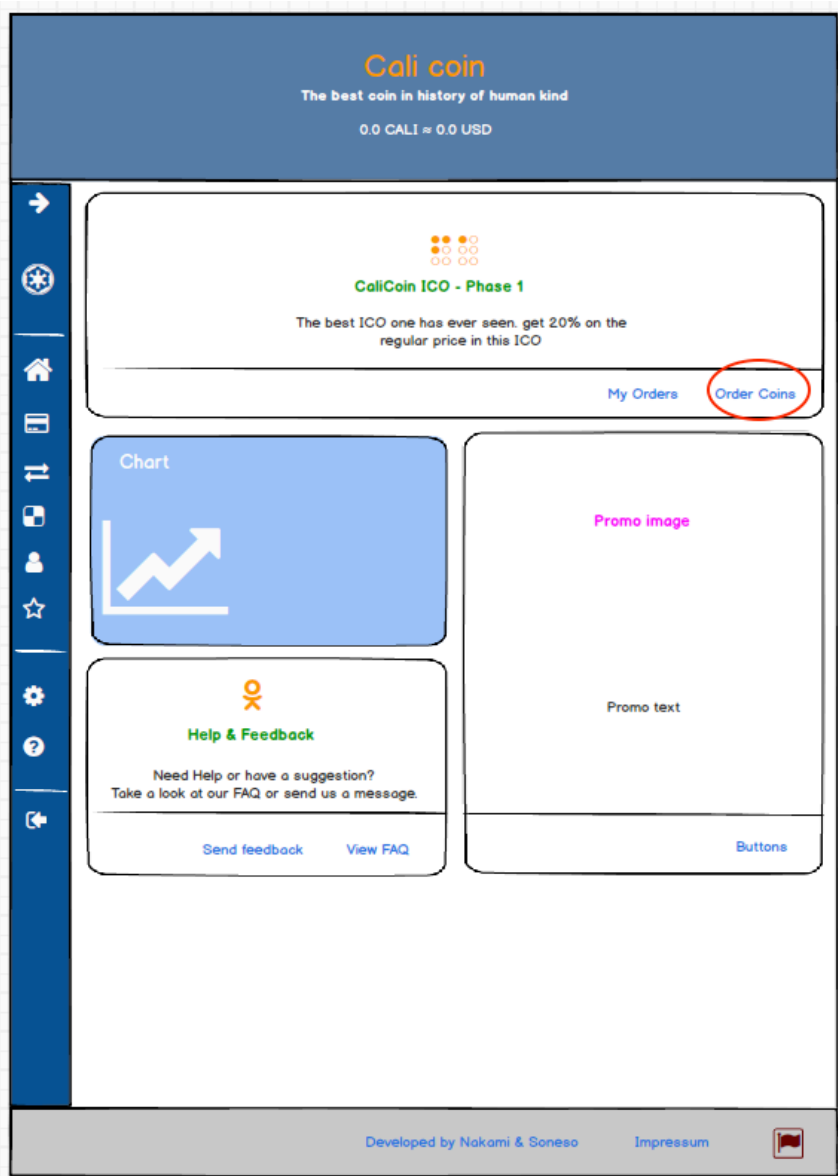


Image 1 –Mockup of web client showing the ICO card and order button

As soon as the user presses the “Order Coins” button, the “New order” form in the expanded area.

Cali coin
The best coin in history of human kind
1.0 CALI ≈ 2.91 USD

CaliCoin ICO - Phase 1
The best ICO one has ever seen. get 20% on the regular price in this ICO

[My Orders](#) [Order Coins](#)

New order ✕

Price for 1 CALI: 0.00125 BTC or 0.0145 ETH or 37.5 XLM

Purchase amount: CALI

Pay with:

You pay: 0.25 BTC

You must transfer the exact amount, otherwise the order can not be executed and your payment will be refunded less any transaction costs. This ICO has a limited number of coins. After these are consumed, no further order can be filled. That's why the rule "First pay, first serve" applies. If you send a payment that can no longer be accepted, it will be refunded less any transaction costs.

I accept ☐

I also accept the cali coin ico terms ☐

Developed by Nakami & Soneso [Impressum](#)

Image 2 –Mockup of web client showing the expanded ICO card for a new order

The view displays the price for one unit of the token in all available currencies, number of tokens left and end date of the ICO Phase. To load the needed data, the client uses the server interface described [here](#).

Below of that, the user can choose how much of the arbitrary coins he would like to purchase. Then, she can select the currency that she would like to use to pay for the coins. Depending on the price per unit, the server automatically calculates how much of the selected currency the user needs to pay. To request the total price, the client uses the server interface described [here](#).

If the user has funded wallets, the client also displays a dropdown, so that the user can select the wallet where he would like to receive the payment. Only funded wallets are shown in the dropdown.

Also, an info and accept checkboxes are displayed. As soon as the user filled the form, she can press the “Order now” button. By doing so, the order is placed. To place the order, the client uses the server interface described [here](#). The parameter for the receiving wallet public key is only to be filled, if the user selected a funded wallet from the dropdown.

When placing the order, the server returns the order object as a result. The client uses it to displays a screen with the summary of the order and deposit address or bank account for the payment.

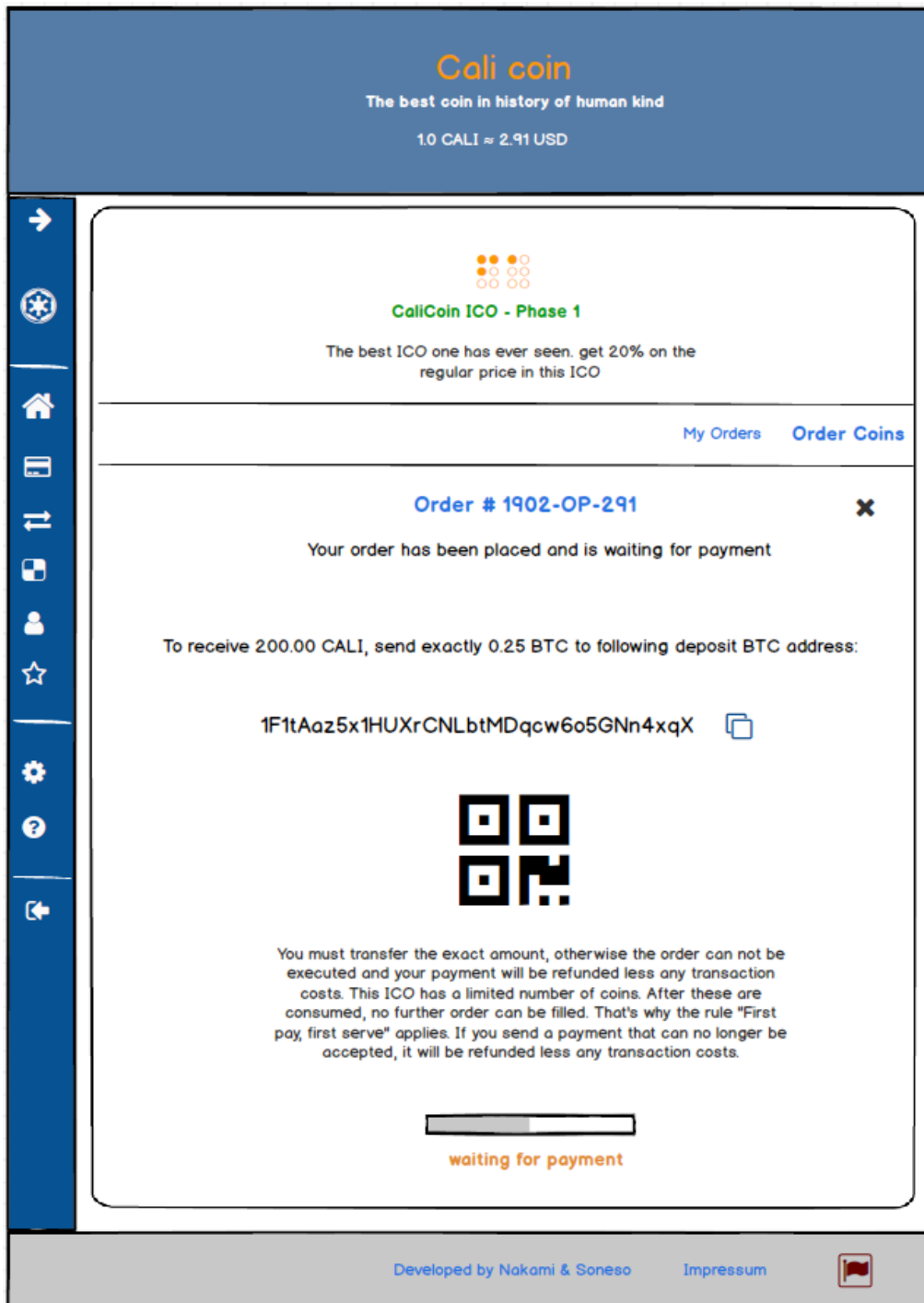


Image 3 –Mockup of web client showing the deposit address for a placed order

It shows the order number in the title and informs the user how to proceed. In the lower part of the screen, a progress bar is displayed, informing the user that the system is waiting for her payment (except for FIAT). Here the client must poll the order details server interface described [here](#) to find out if the status of the order has changed (e.g. because the user transferred the payment, or some error occurred). In case of FIAT no progress bar is displayed, no polling here.

If the user sends the crypto-payment, the server will find out that the payment has been received on the deposit address. It will validate the received amount by comparing it to the amount from the order. If the amounts do not match, the server changes the status of the order, than the client receives the new status and related data in the next poll. The client then stops polling and informs the user that the order could not be completed and the received amount has been refunded.

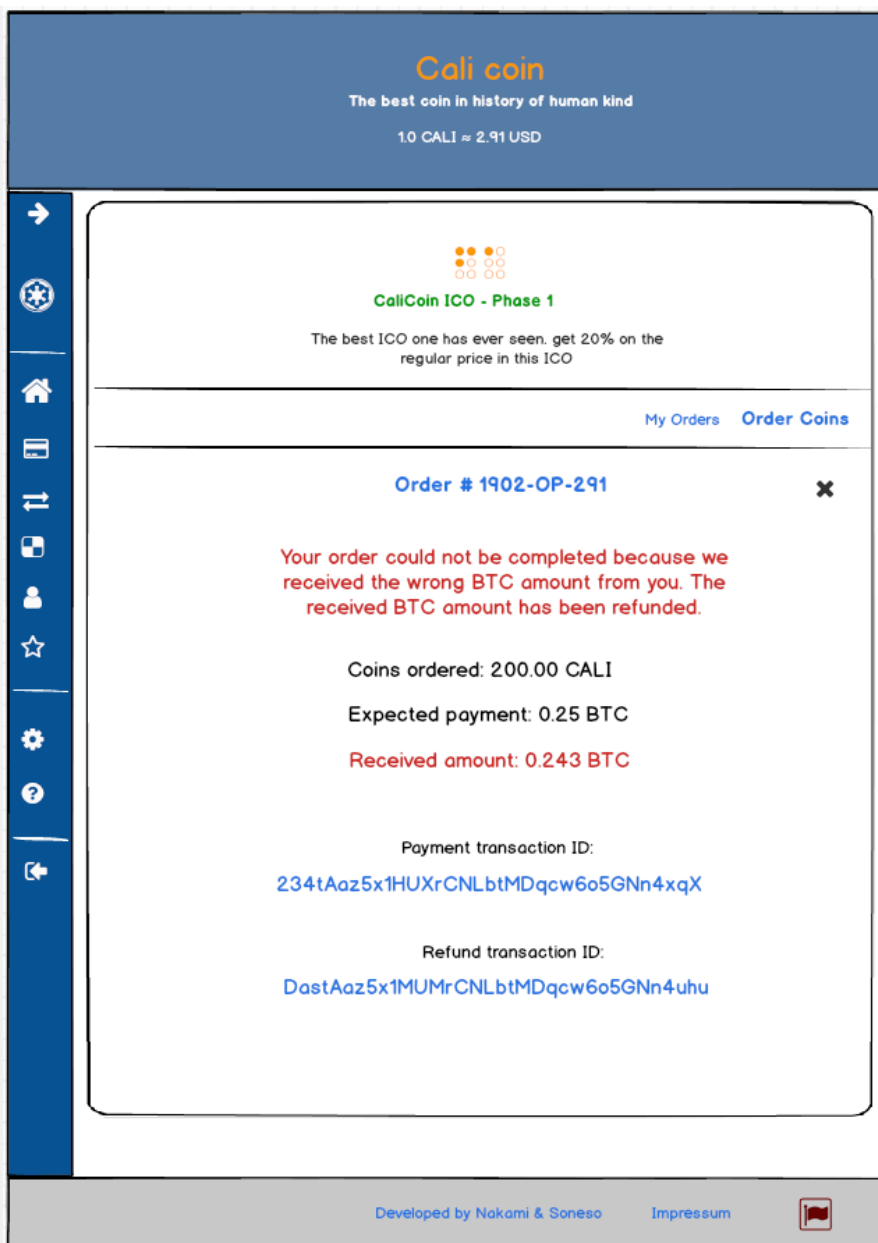


Image 4 –Mockup of web client showing the deposit address for a placed order

The client displays the transaction IDs of the payment and refund as a link, so that the user see the error and refund in the corresponding blockchain.

If the order cannot be filled because all coins to distribute have already been consumed before the user sent his payment, the server changes the status of the order and the client receives the information in the next poll. The client stops polling and informs the user about the rejected order.

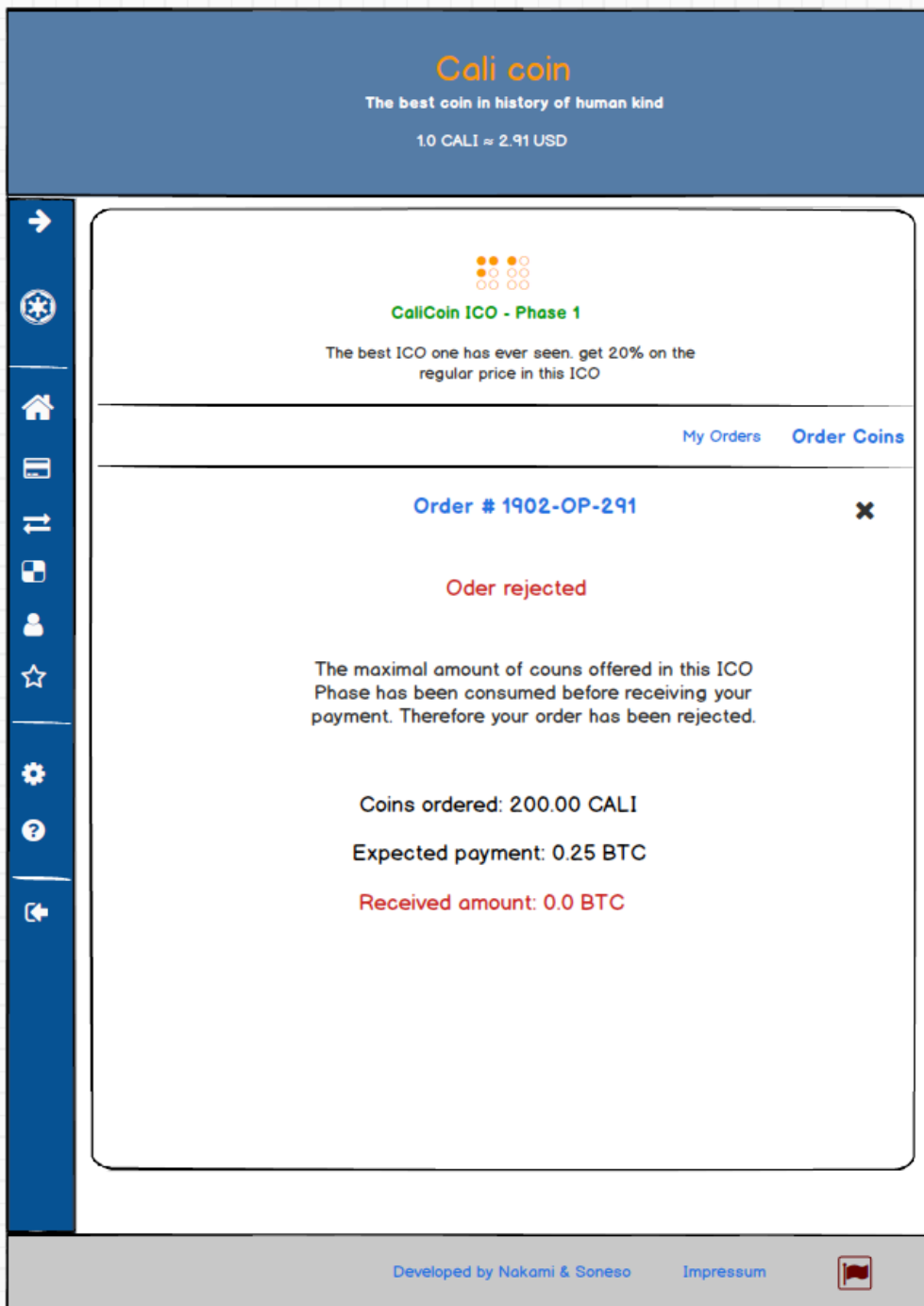


Image 5 –Mockup of web client showing the unpaid order rejected

There is also the case, that the payment has been received, but the server could not complete the order because all coins to distribute have already been consumed. In this case, the server returns corresponding data at the next client poll, so that the client can display the refund transaction.

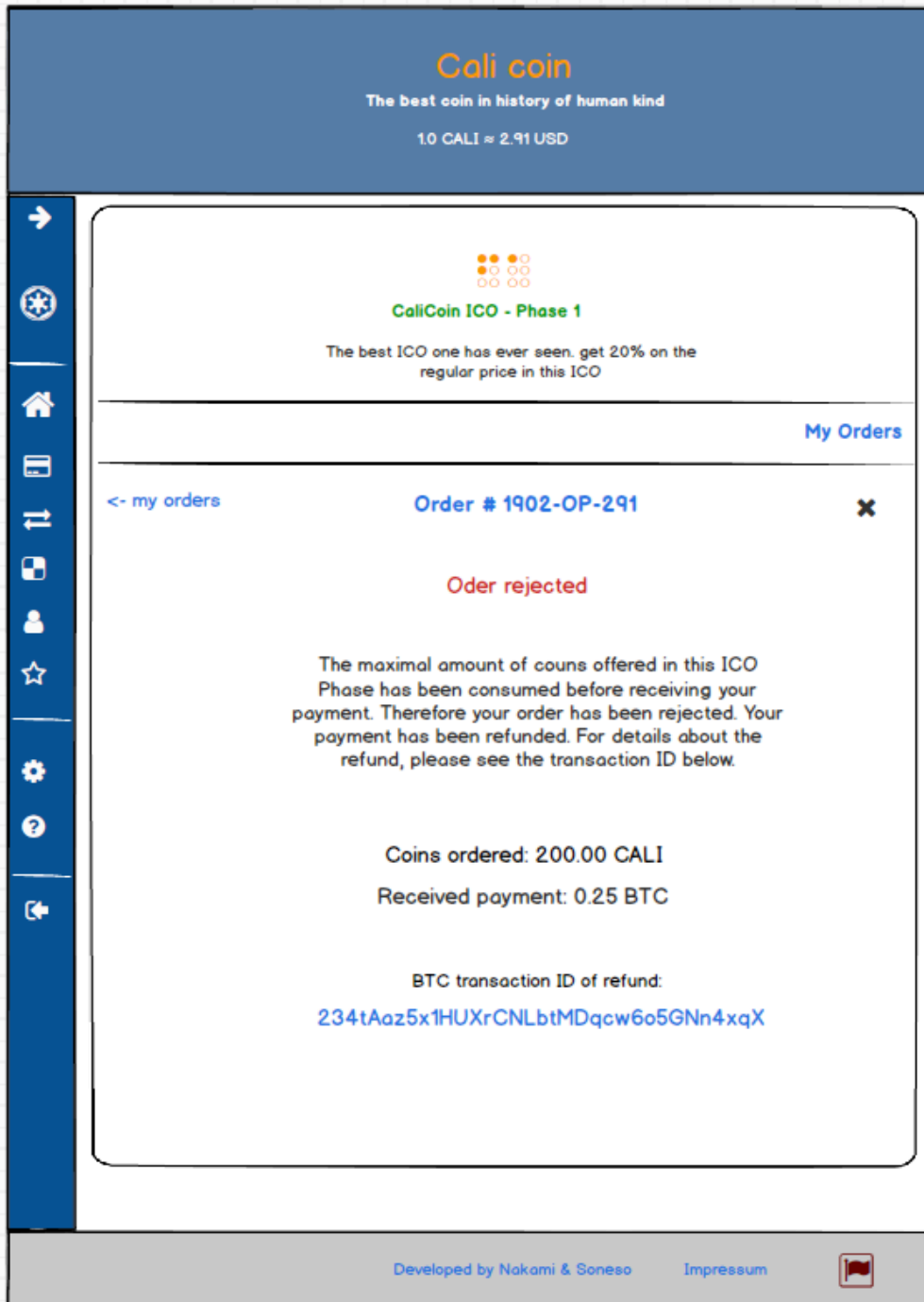


Image 6 –Mockup of web client showing the payed order rejected

The next possible error case is that the ICO Phase ended before receiving the payment from the user.

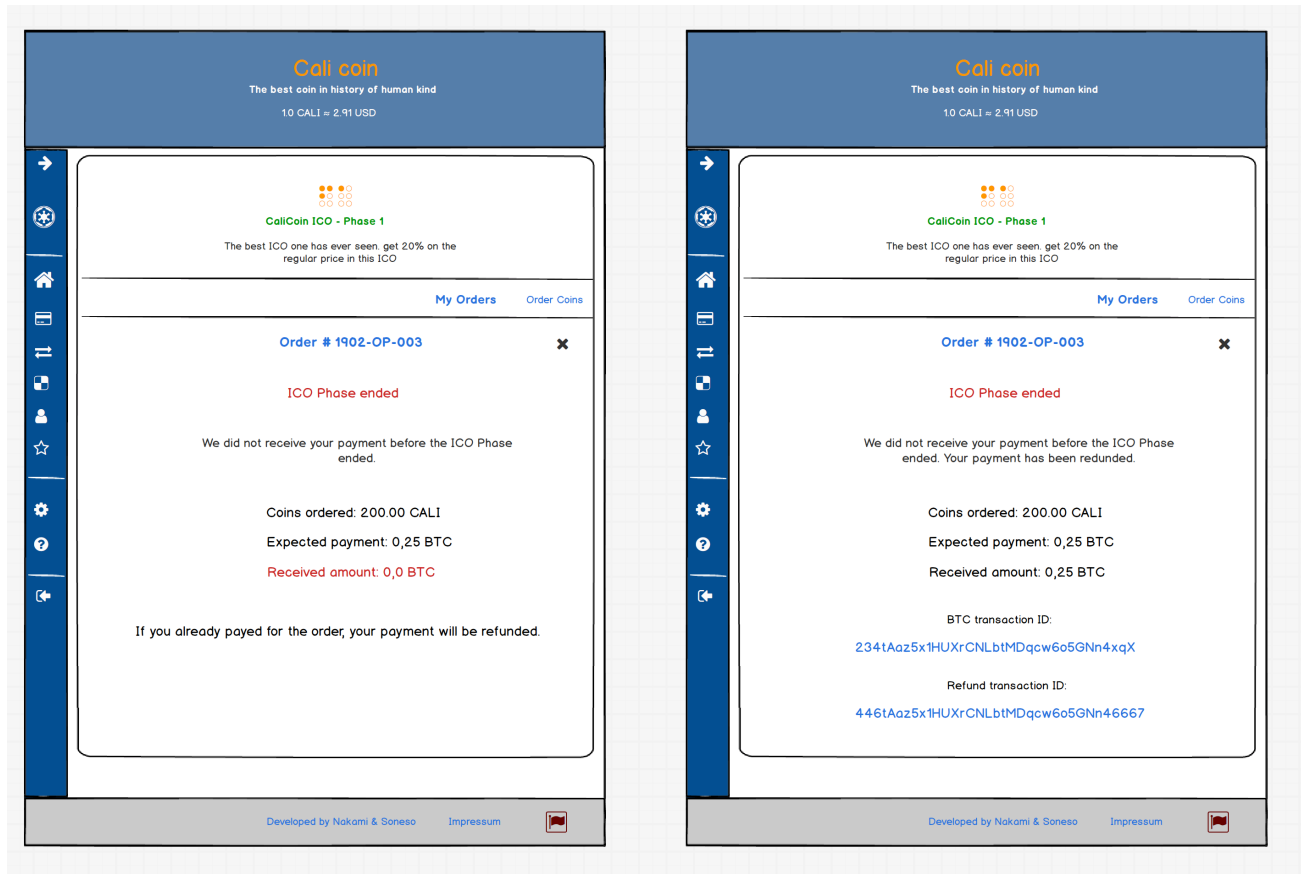


Image 7 –Mockup of web client showing the ICO Phase ended

If the user transferred the correct amount, the server will first change the status to “payment received” and is going to prepare the order for the next steps. In this case the client must continue to poll until the next status update occurs.

After preparing the order, the server changes the status to “waiting for user transaction”. The client stops polling and calls the “get_payment_transaction” server interface to receive the payment transaction for signing. The interface is described [here](#). After receiving the transaction, the client shows the “payment received” screen, so that the user can insert his password and the client can sign the transaction and send back to the server. (TODO multiple signers & master key not enough weight)

By the payment received screen, the client inform the user about the status and requests his password needed for the next step.

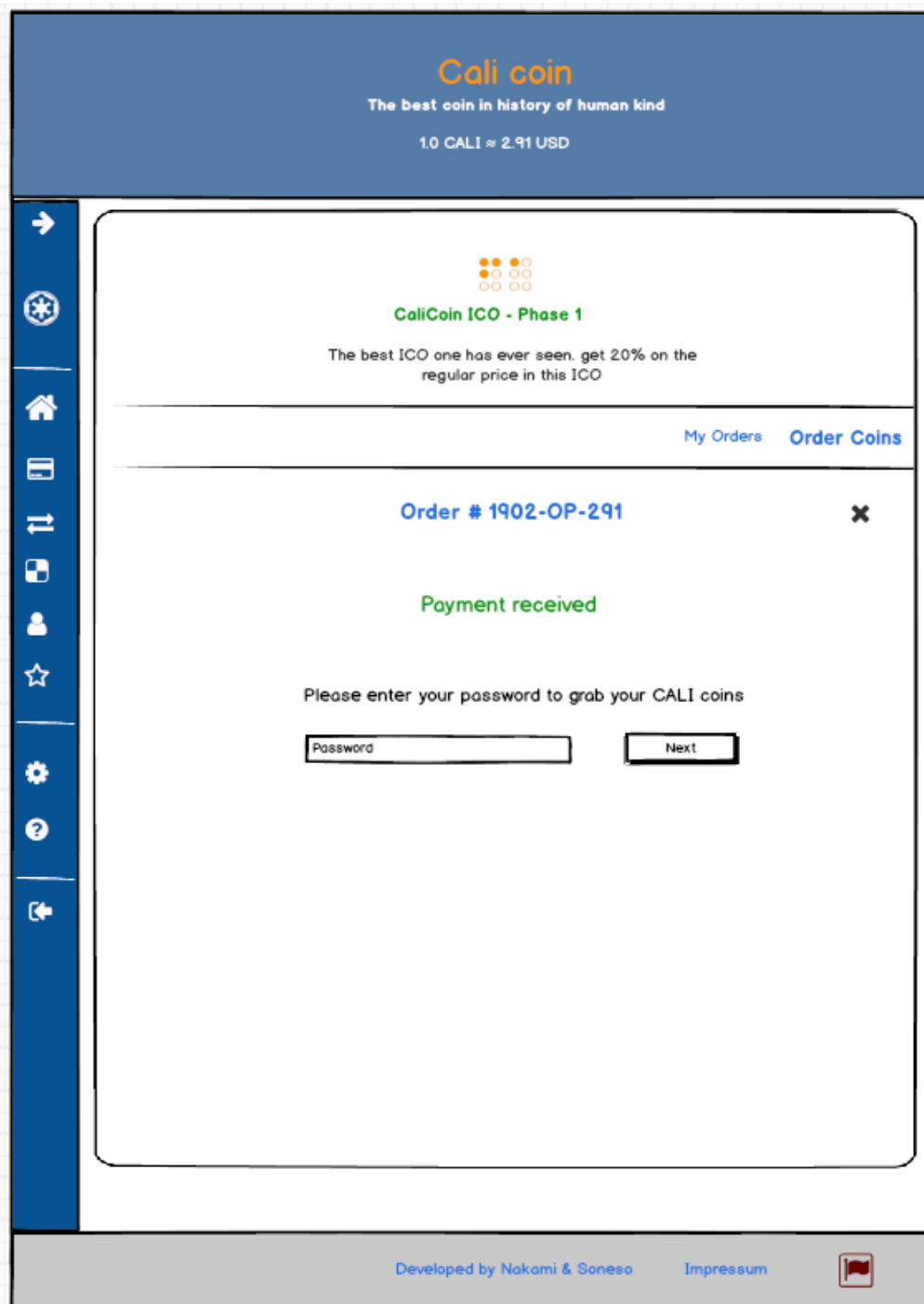


Image 8 –Mockup of web client showing the payment received screen

As soon as the user enters his password and presses the “Next” button, the client can sign the payment transaction and send it back to the server by using the interface “execute_payment_transaction”. The interface is described [here](#). Before doing so it displays the “Requesting” screen.

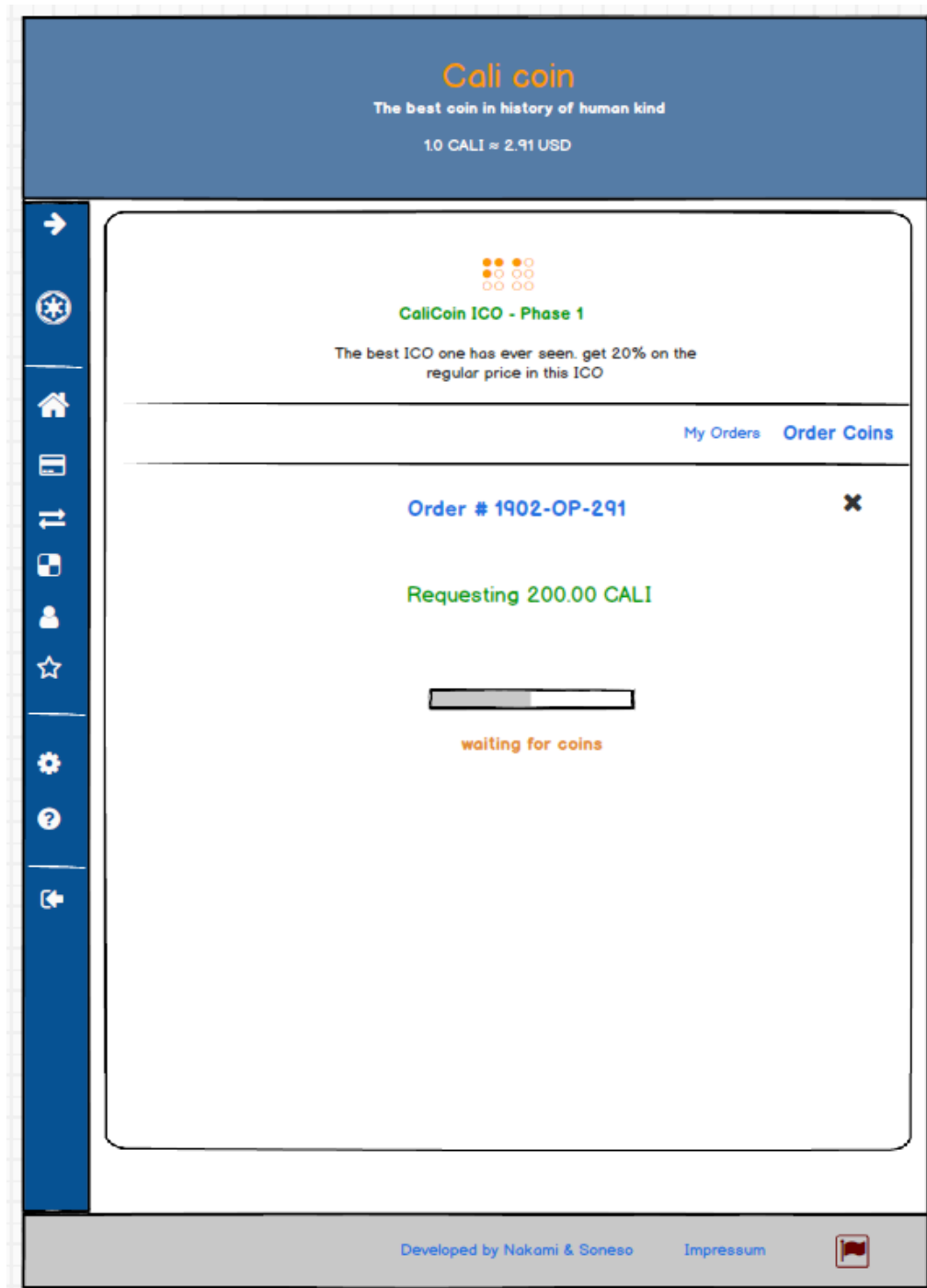


Image 9 –Mockup of web client showing the requesting coins screen

The server in turn will validate the transaction and send it to the stellar network. It responds with success, returns the order object that now has the status “completed”. (what about the validation error case, e.g. account already created?). The client next checks the balance of the wallet and as soon as the coins are available it displays the order successfully completed screen.

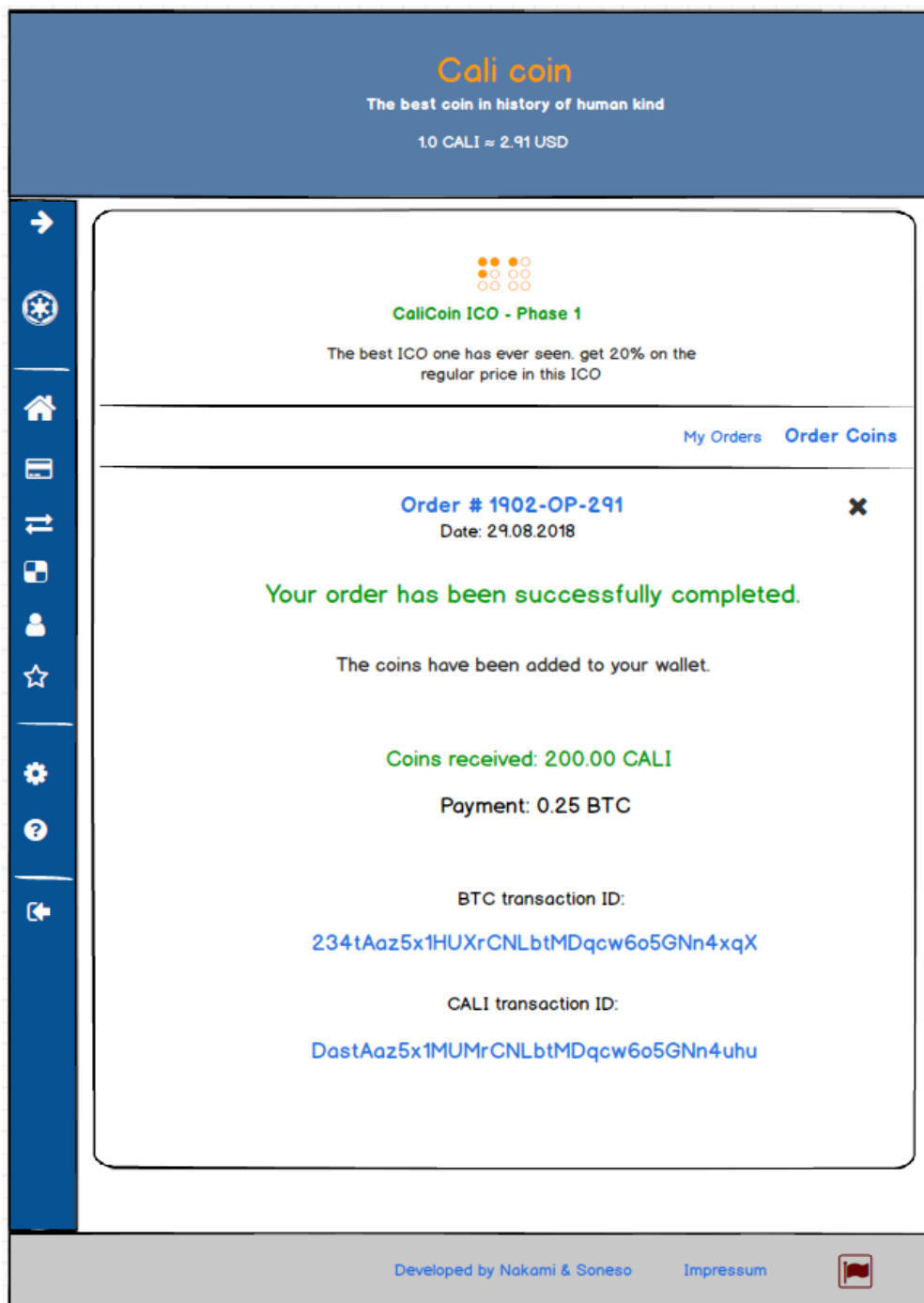


Image 10 –Mockup of web client showing the order completed screen

The client displays the order details including the transaction ids as links so that the user can always see them in the corresponding blockchain.

5. Viewing existing orders

5.1 Overview

The user can always see his orders by pressing the “My Orders” button in the ICO wallet card. As soon as pressed, the ICO card will expand and will show the list of orders placed by the user. The client receives the data from the interface described [here](#).

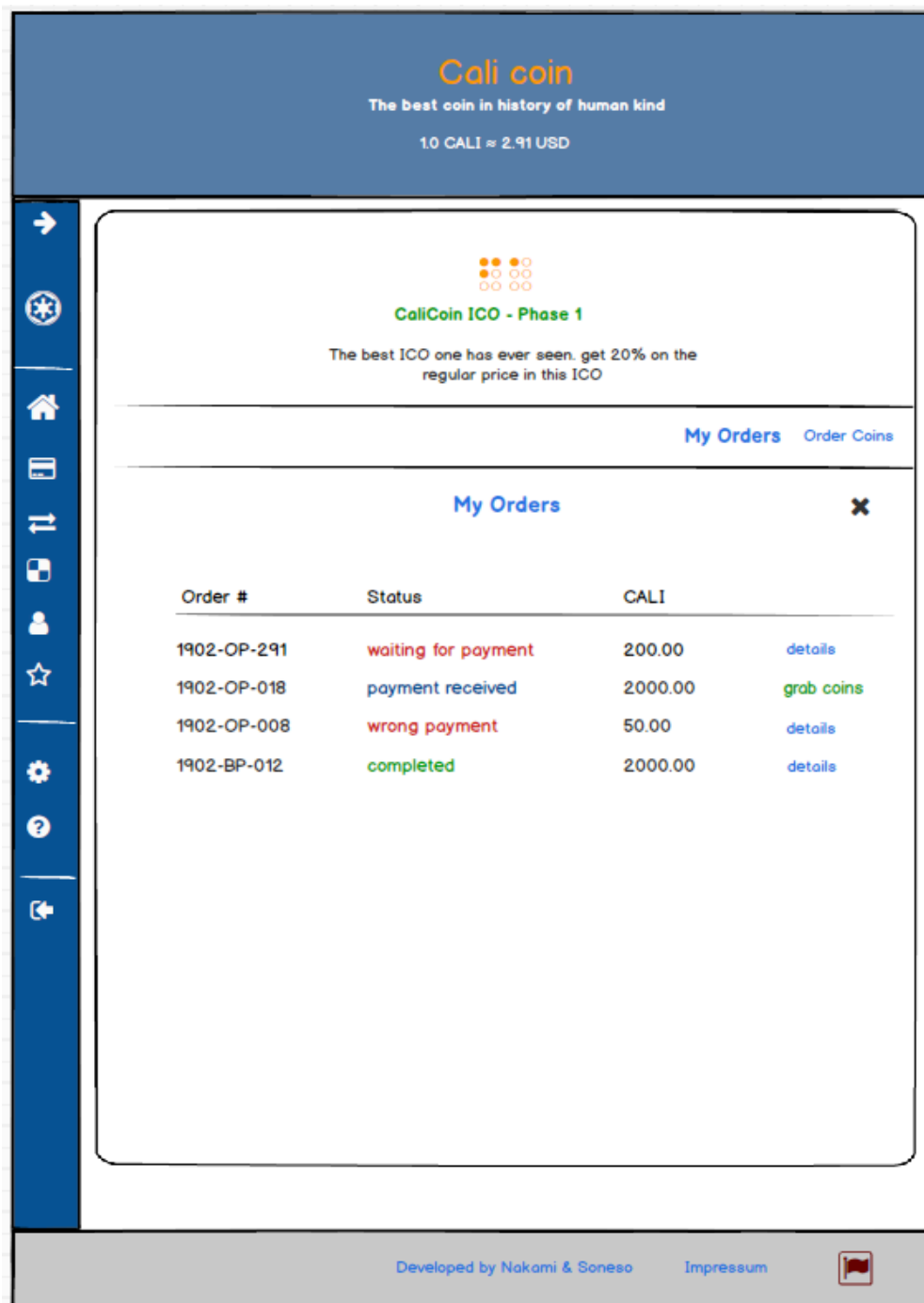


Image 11 –Mockup of web client showing my orders screen

The table shows all orders, displays their status, amount of coins ordered and an action button in each row. Following status are possible to be shown to the user:

- Waiting for payment (order status: waiting for payment or payment received)
- Payment received (order status: waiting for user transaction)
- Wrong payment (order status: wrong amount received)
- Completed (order status: completed)
- Rejected (order status: order rejected – phase ended/coins consumed before payment)

The client displays a “details” action button in each row, except when the status of the row is “Payment received”. In this case, it displays a “grab coins” action button.

If the user presses the “details” button, the details of the order will be displayed in a popup (see screens above).

If the user presses the “grab coins” action button, the client displays the “payment received” screen in a popup (see above) and applies the logic described above to grab the users coins.

If the user presses no button, the client polls the server interface for the order list to be able to update the status of the orders if something changed.

6. Processing orders at login

When the user logs in, the client checks the field “payment_state” in the user profile. If the payment state is set to open, the client request the list of orders with the status “waiting for user transaction” from the server. It uses the server interface for the order list described above to do so (+ parameter order status). If the returned list is empty, the user will be redirected to the dashboard. Otherwise, If the list is not empty the client will display a processing screen (like wallet setup) and will immediately start processing each order from the list (logic as described above, client already has user password from login). The client will process the orders sequentially, showing following screen:

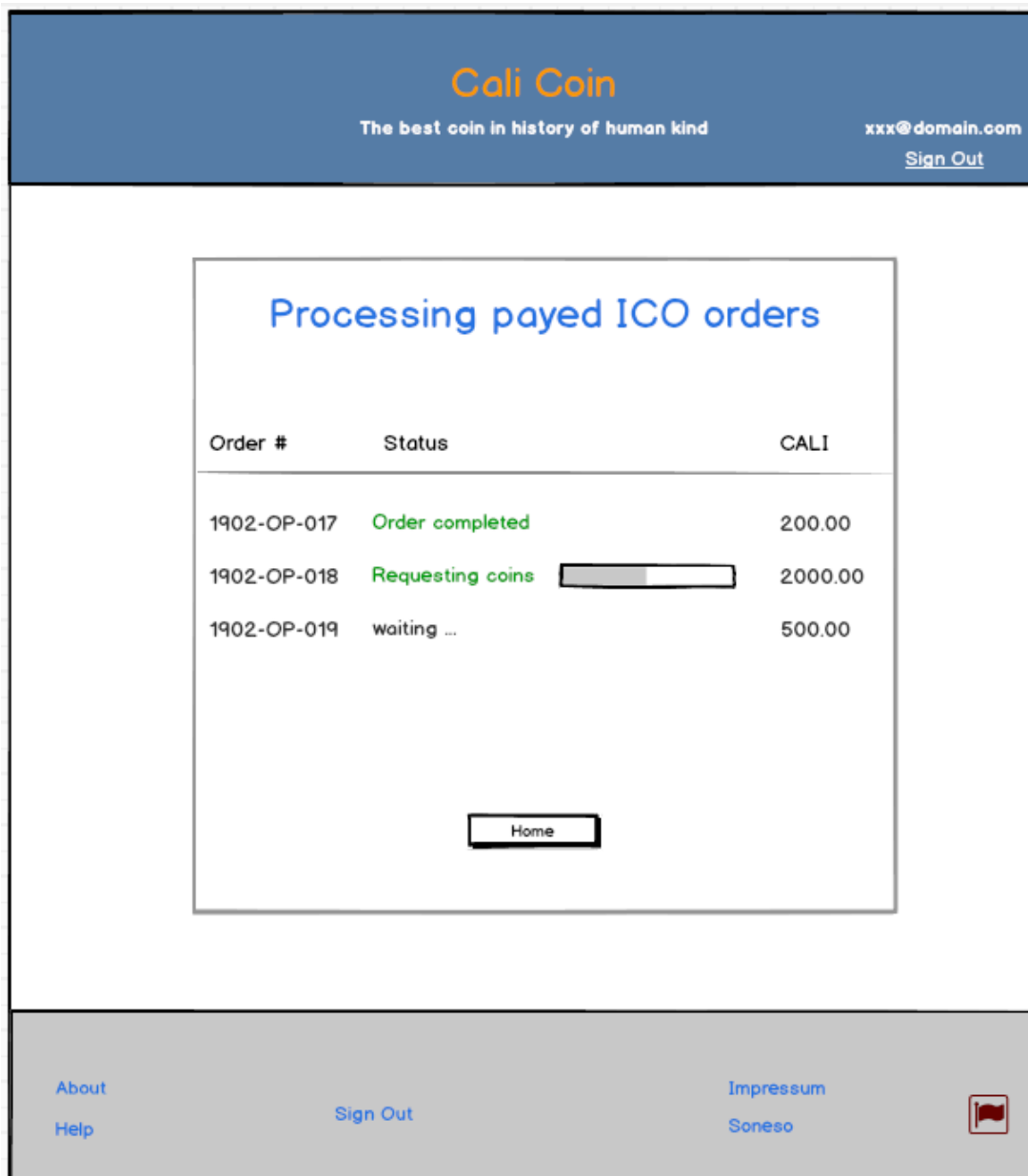


Image 12 –Mockup of web client showing the details of an order with status “completed”

As soon as an order is completed it will mark it as completed. While waiting to be processed the orders in the list are marked as “waiting ...”. As soon as the system finished processing all orders from the list it will automatically redirect the user to the dashboard. If the user doesn’t want to wait for the client to process the orders, she can press the “Home” button. The client will interrupt the process and redirect the user to the dashboard. (TODO: multisig, master not enough weight)

7. Server Interfaces

7.1 Details for ICO Phase

parameter: ico_phase_id

return:

- list of activated currencies for this ico phase including price per token

-- exchange_currency_id, asset_code, price_per_token,

7.2 Calculated payment sum

parameter: ico_phase_id, amount, exchange_currency_id

7.3 Create Order

parameter: ico_phase_id, amount ico token, exchange_currency_id, total_price (received from 2)

return: order_id, status of order, amount of tokens ordered, asset code of token,

total price in exchange currency,

asset code of exchange currency, exchange_currency_type,

deposit_pk, qr_code (only for crypto), bank data (only for fiat)

7.4 Order details

4. order_details

parameter: order_id

returns: order_status, and related data

4.a) status: wrong amount received,

- amount of tokens ordered

- asset code of token

- exchange currency type
- asset code of exchange currency
- expected payment sum in exchange currency
- received payment amount in exchange currency
- user payment transaction id
- refund transaction id

4.b) status: order rejected

- case 1: phase ended before payment

like 4a) + date of payment, + end date of ico phase

- case 2: coins consumed before payment (no coins fucking left)

like 4 a) + date of payment

4 c) status: waiting for payment

- order_id, status of order, amount of tokens ordered, asset code of token, total price in exchange currency, asset code of exchange currency, exchange_currency_type, deposit_pk, qr_code (only for crypto), bank data (only for fiat)

4 d) status payment received:

- continue polling

4 e) waiting for user transaction

-> client: shows to user payment received view

-> client checks if user account exists and is funded with enough lumen to add trustline

if account exists and trustline exists than no need to see if enough lumens

-> NO: client calls interface "fund_account"

-> YES: if trustline does not exist => request pwd from user to add trustline, create trustline

if trustline exists (or after created):

-> client: request user transaction from server interface: "get_payment_transaction"

7.5 Get payment transaction

parameter: order_id

return: transaction

-> client (request pwd - if not already) sign transaction received from server

-> send back to server using interface: "execute_payment_transaction"

7.6 Execute payment transaction

parameter: signed transaction

return: success (+ order details, status is now "completed" + stellar ico token transaction id)

7.7 Get order list

param optional ico_phase, order_status

to get the order details call the poll interface

8. Images

Image 1 –Mockup of web client showing the ICO card and order button	6
Image 2 –Mockup of web client showing the expanded ICO card for a new order	7
Image 3 –Mockup of web client showing the deposit address for a placed order	8
Image 4 –Mockup of web client showing the deposit address for a placed order	9
Image 5 –Mockup of web client showing the unpaid order rejected	10
Image 6 –Mockup of web client showing the payed order rejected.....	11

Image 7 –Mockup of web client showing the ICO Phase ended	12
Image 8 –Mockup of web client showing the payment received screen.....	13
Image 9 –Mockup of web client showing the requesting coins screen	14
Image 11 –Mockup of web client showing the order completed screen.....	15
Image 12 –Mockup of web client showing my orders screen.....	16
Image 13 –Mockup of web client showing the details of an order with status “completed”	18