Use cases

This section describes a detailed description of the processes provided in the QR Payments solution and the appearance of the application from the end user point of view.

Wallet Server Money Transfer API

This section describes use cases which can be initiated using Wallet Server P2P API. This API should be used by Customers through integrated Wallet QR SDK to manage Transactions and Transaction’s Senders/Receivers, Commission calculation and determine Currencies on Wallet Server. Every method below is secured by session token. *For more detailed information about non-QR functionalities offered by Money Transfer please see Money Transfer documentation.*

Transaction process

If the user and his card are present in the Verestro system, such a user can perform QR transactions using the Verestro QR Payments Hub solution. This type of transaction allows the user to make a transfer by scanning the QR code and to track the transfer by providing his own code.

QR Payment - scan code

This diagram shows QR code transaction processes in the case that the user scans the "external" QR code:

@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold

```
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Verestro THC API" as thc
participant "Acquirer" as acq
participant "Issuer" as acs
user->mob: 1. User scans QR code
mob->sdk: 2. Provides scanned QR code
sdk->sdk: 3. Parses obtained QR - extract data
note right of sdk: Wallet SDK uses dedicated library for QR handling - IMV Standard
mob<-sdk: 4. Returns extracted data from scanned QR code
user<-mob: 5. Shows all transaction data extracted from QR code - optional
note right of user: If QR code is static, user provides amount
user->mob: 6. Accepts transaction
note right of user: 3ds authentication may be required at this point
note right of user: 3ds authentication flow described on above diagram
mob->sdk: 7. Performs transaction
sdk->p2p: 8. Provides transaction data and execute transaction
p2p->acq: 9. Perform transaction
acq->acs: 10. Perform Funding from provided card if possible
acq<-acs: 11. Success
p2p<-acq: 12. Success + transaction id
p2p->thc: 13. Store transaction with status Funding
p2p<-thc: 14. Transaction has been stored successfully
sdk<-p2p: 15. Transaction success
mob<-sdk: 16. Transaction success
user<-mob: 17. Your transaction has been sent
@enduml
```

After scanning the received QR code the Mobile SDK decomposes the data contained in the QR code. The user sees the transaction data, such as the amount that will be sent after confirming the transfer or the currency in which the transfer will be made etc. After confirming the transfer of the funds, the SDK forwards the request to the Wallet API. Wallet API in turn forwards the request to the Acquirer. Based on the data received, the Acquirer contacts the bank to complete the transaction. If the bank agrees, the funding process is carried out - collecting funds from the Sender's account. After the funding is completed, the funds are transferred to the appropriate Receiver.
To facilitate understanding of the process flow, each of the steps is described below in the correct

order (points highlighted in blue are performed outside the Verestro Server):

1. User scans QR code using device (possible only if provided the device supports such functionality).
2. Verestro Mobile SDK gets data from the scanned QR code.
3. User confirms the transfer and authenticates himself via 3ds authentication process.
4. After going through the 3ds process, the funds transfer process begins.
5. The application orders the transfer of funds by communicating with the Verestro backend via the Verestro Mobile SDK.
6. Money Transfer API communicates with Data Core to check whether the card details belong to the user and the receiver.
   - After receiving a positive response from Data Core, Money Transfer API performs money send transaction.
7. Acquirer receives a request to make a transfer of funds.
8. The Acquirer communicates with the Issuer regarding the execution of the transfer.
   - The Sender's account is debited.
   - The Receiver’s account is credited.
9. The Acquirer receives information from the Issuer that the operation has been performed.
10. Acquirer informs Money Transfer API about the positive status of the ordered operation.
11. Money Transfer API saves transaction details in the Transaction History Core API.
12. Money Transfer API informs Mobile SDK about the positive status of the ordered operation.
13. Mobile SDK informs Mobile Application about the positive status of the ordered operation.
14. The Mobile Application displays to the user a successful transfer of funds.

QR Payment - generate code

This diagram shows QR code transaction processes in the case that the user generates QR code:

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
```

```
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Payer" as payer
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Verestro THC API" as thc
participant "Acquirer" as acq
participant "Issuer" as acs
user->mob: 1. User opens QR code generating
note right of user: User generates QR
mob->sdk: 2. Provides data fulfilled by the user
sdk->sdk: 3. Coverts obrained data into QR code
note right of sdk: Wallet SDK uses dedicated library for QR handling - IMV Standard
mob<-sdk: 4. Returns QR with parsed data
user<-mob: 5. Shows all transaction data and generated QR code
note right of user: User shows generated QR to the potential payer
note right of payer: Payer scans QR code
payer->payer: 6. Scans QR code
mob->sdk: 7. Get transaction data from QR code
mob<-sdk: 8. Returns transaction data obtained from QR code
payer<-mob: 9. Shows all transaction data
payer->mob: 10. Confirms money transfer and pass 3ds
note right of payer: 3ds authentication may be required at this point
note right of payer: 3ds authentication flow described on above diagram
mob->sdk: 11. Performs transaction
sdk->p2p: 12. Provides transaction data and execute transaction
p2p->acq: 13. Perform transaction
acq->acs: 14. Perform Funding on Payer's card if possible
acq<-acs: 15. Success
p2p<-acq: 16. Success + transaction id
p2p->thc: 17. Store transaction with status Funding
p2p<-thc: 18. Transaction has been stored successfully
sdk<-p2p: 19. Transaction success
mob<-sdk: 20. Transaction success
payer<-mob: 21. Your transaction has been sent
acq->acs: 22. Perform Credit on User's card if possible
acq<-acs: 23. Success
p2p<-acq: 24. Success + transaction id
p2p->thc: 25. Update transaction status to CLEARED
@enduml
```

3DS Authentication

The QR Money Transfer Hub Solution supports the 3DS 2.0 process and it is required when user is initiating a transaction. This is an authentication method based on the alleged cardholder data check, biometric authentication and improved customer experience.

As mentioned in the "Configuration" paragraph, a specific 3DS integration is required depending on which ACQ Verestro will connect to when making a transaction.

- If the integration with a given ACQ has already been made, Verestro only need to configure the appropriate merchant identifier, which should be provided by the Customer.

If the integration with the required billing agent has not yet been completed, it will be one of the activities for which Verestro will be responsible. Note that integration with 3DS increases the scope of required development.
In card to card transfer, the authentication process is required. It is to confirm that the user is definitely the owner of the card.

This diagram shows high level 3ds authentication processes:

```
@startuml
skinparam ParticipantPadding 30
skinparam BoxPadding 30
skinparam noteFontColor #FFFFFF
skinparam noteBackgroundColor #1C1E3F
skinparam noteBorderColor #1C1E3F
skinparam noteBorderThickness 1
skinparam sequence {
ArrowColor #1C1E3F
ArrowFontColor #1C1E3F
ActorBorderColor #1C1E3F
ActorBackgroundColor #FFFFFF
ActorFontStyle bold
ParticipantBorderColor #1C1E3F
ParticipantBackgroundColor #1C1E3F
ParticipantFontColor #FFFFFF
ParticipantFontStyle bold
LifeLineBackgroundColor #1C1E3F
LifeLineBorderColor #1C1E3F
}
participant "User" as user
participant "Mobile App" as mob
participant "Verestro Mobile SDK" as sdk
participant "Verestro Money Transfer API" as p2p
participant "Acquirer" as acq
participant "Mastercard/VISA" as mcvisa
participant "ACS/Issuer" as acs
note left of mob: User has confirmed currencies and accepted shown currency rate
```

user->mob: 1. Enters CVC code
note left of mob: In this step the application begins 3DS authentication process.
note left of mob: If the bank decides that 3ds is not required, not any action will be performed.
note left of mob: Diagram shows the option requiring the user to authenticate.
mob->sdk: 2. Initialize 3DS process
sdk->p2p: 3. Initialize 3DS process
p2p->acq: 4. Initialize 3DS process
note left of acs: Bank returns decision whether it is necessary for the user to authenticate
acq->mcvisa: 5. Is 3DS authentication required?
acq<-mcvisa: 6. ThreeDS Method
p2p<-acq: 7. ThreeDS Method
p2p->acq: 8. Continue 3DS process
acq->acs: 9. Continue 3DS process
acq<-acs: 10. Challenge required
p2p<-acq: 11. Challenge required
sdk<-p2p: 12. Challenge required
mob<-sdk: 13. Challenge required
user<-mob: 14. Informs user that challenge is required
note left of mob: Bank's page content is provided
user->user: 15. Performs challenge
note right of user: After a successful challenge, the Bank sends PaRes/cRes, which is intercepted by the Wallet SDK and provided to the Money Transfer API
user-->mob
mob-->sdk
sdk->p2p: 16. Provides intercepted PaRes/cRes
p2p->acq: 17. Provides obtained PaRes/cRes
acq->acs: 18. Check authentication for provided PaRes/cRes
acq<-acs: 19. Authentication success
p2p<-acq: 20. Authentication success
sdk<-p2p: 21. Authentication success
mob<-sdk: 22. Authentication success
user<-mob: 23. Informs about the successful completion of the authentication process
@enduml

To facilitate understanding of the process flow, each of the steps is described below in the correct order *(points highlighted in blue are performed outside the Verestro Server)*:

1. Mobile application contacts the Verestro Server via the Verestro Mobile SDK to start 3ds process.
2. Mobile SDK provides all necessary data to the Money Transfer API (including user/card id and 3ds authentication request id) while calling the 3ds initialization method.
3. Having all the necessary information, Money Transfer API orders Acquirer to start the 3ds authentication process.
4. Acquirer transfers the card and user details to ACS.

5. If the user is the owner of the card, the ACS returns a positive answer and a decision whether the continuation of the authentication process is required (according to the diagram above, the scenario with the necessity to continue the process is described).
6. ACS informs the Acquirer that the 3ds process continue is required.
7. Acquirer informs Money Transfer API about ACS decision.
8. Money Transfer API requests Acquirer to continue the authentication process (the authentication id is provided.
9. Acquirer provide the request to continue the process to the ACS.
10. ACS informs about the necessity to perform the Challenge process and returns necessary parameters such as:
    - challengeHtmlFormBase64 - this field is a BASE64 encrypted html source file containing the ACS' challenge 3DSecure frame.
    - cReq - data for building a form such as challengeHtmlFormBase64.
11. Acquirer informs Verestro Money Transfer API that ACS requires Challengeand provides above parameters.
12. Verestro Money Transfer API forwards the obtained information to Verestro Mobile SDK.
13. Verestro Mobile SDK decodes the received challengeHtmlFormBase64 parameter and transmits the received frame of the mobile application.
14. The user is redirected to the bank's website where he performs the Challenge process.
15. After a successful Challenge process, the bank sends the cRes / PaRes parameter. This response is intercepted by the Verestro Mobile SDK and forwarded to the Verestro Money Transfer API.
16. Verestro Money Transfer API provides the received cRes / PaRes to Acquirer.
17. Acquirer provides the above parameters to ACS for verification.
18. If everything was done correctly, the ACS informs Acquirer about the successful completion of the 3ds authentication. Among other things, the following parameters are included in the response.
    - cavv - property determined by the ACS. The value may be used to provide proof of authentication.
    - eci - property is determined by the ACS. This property contains the two digit Electronic Commerce Indicator (ECI) value, which is to be submitted in a credit card authorization message.
19. Acquirer provides information about successful completion of the 3ds authentication among with the above parameters obtained from the ACS to Verestro Money Transfer API.
20. Verestro Money Transfer API provides information about successful completion of the 3ds authentication among with the above parameters obtained from the ACS to Verestro Mobile SDK.
21. Verestro Mobile SDK provides positive response to mobile application. The information is shown to the user.

User experience & screenshots

This chapter introduces the main actions and processes in the application associated with QR module from the end

user point of view.

> *The rest of the processes apply to "Money Transfer" as a product as a whole are described in separate Money Transfer documentation. QR payments is one of the components of the Money Transfer product.*

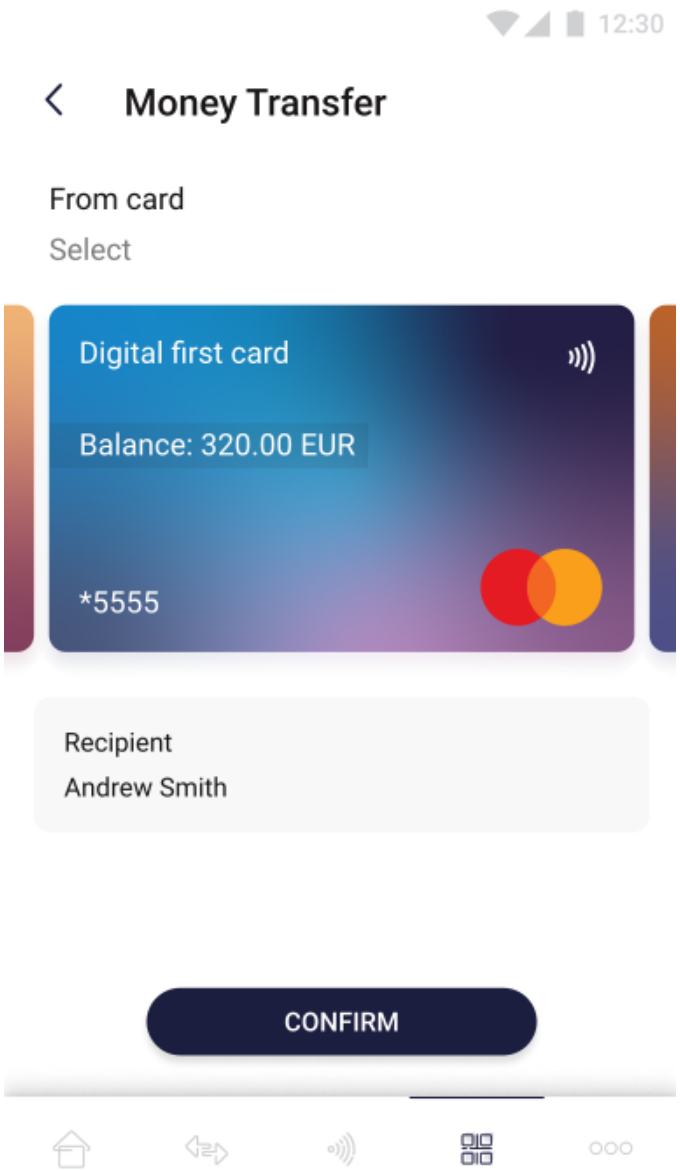
QR Payment flow

The module responsible for QR payments can be divided into two functionalities. The first one allows you to make a payment by scanning the QR code. The second functionality allows user to generate a QR code with which user can order a transfer to his card.

Scan QR code

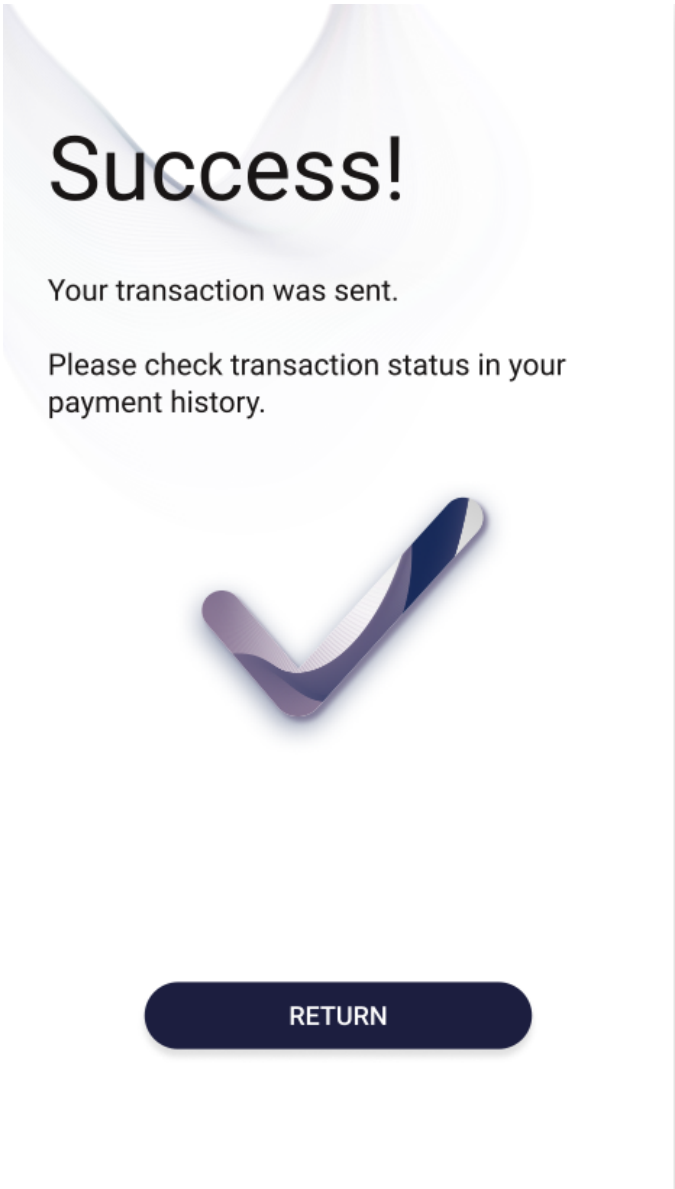
The scan QR option allows user to order a money transfer by scanning the QR code issued by the potential recipient. The execution of this path depends on whether the user's device has a QR scan function. Additionally, the application may request access to the camera of the device used.

The pictures below show “scan QR code” view:



On the screen above the user selects the scan tab for the application to get all the information needed to make the transfer from the provided QR code.

On the screen above the redirection following scanning the QR code is shown. The user can see to whom the transfer will be made and which of his cards will be debited. If the user is ready to make the transfer, he confirms the action with the confirm button. Similar to a card to card transaction, the 3ds process may be required (See “Card to card” chapter).



On the screen above user is informed about the successfully money transfer. Return button will redirect the user to the main page.

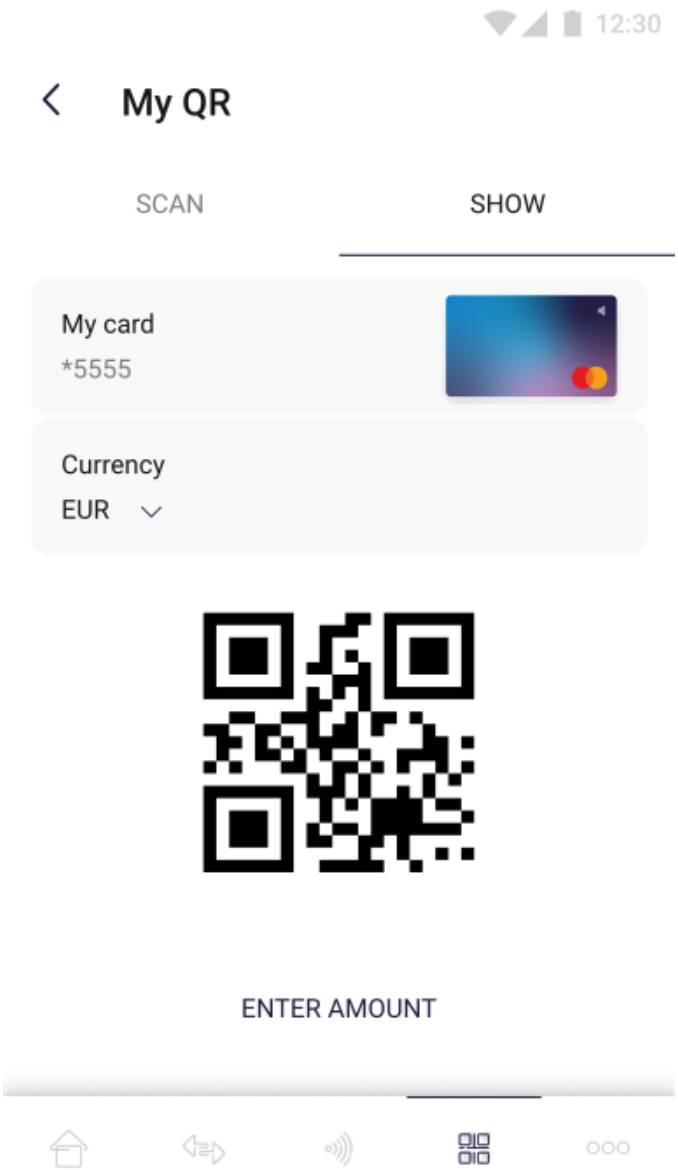
On the screen above user is informed about the unsuccessfully money transfer. In this case user is able to try again the transfer or return to the main page.
Example reasons of the transaction fail:

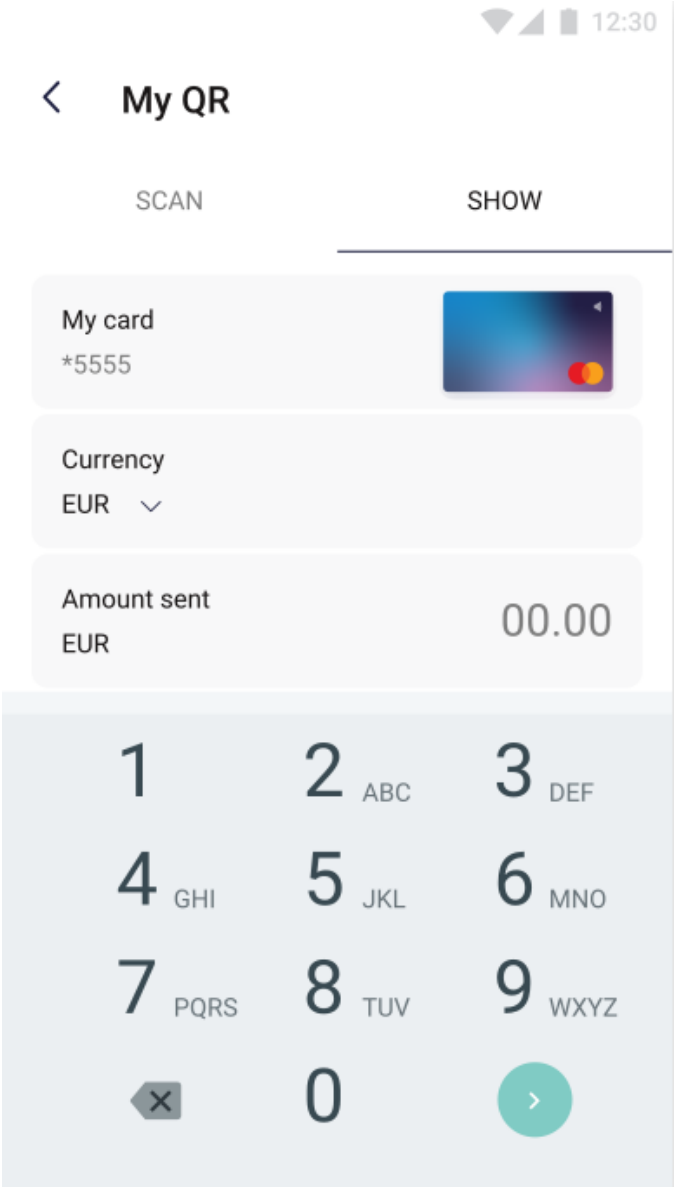
- Luck of funds,
- 3ds failed,
- Invalid card data.

Generate QR code

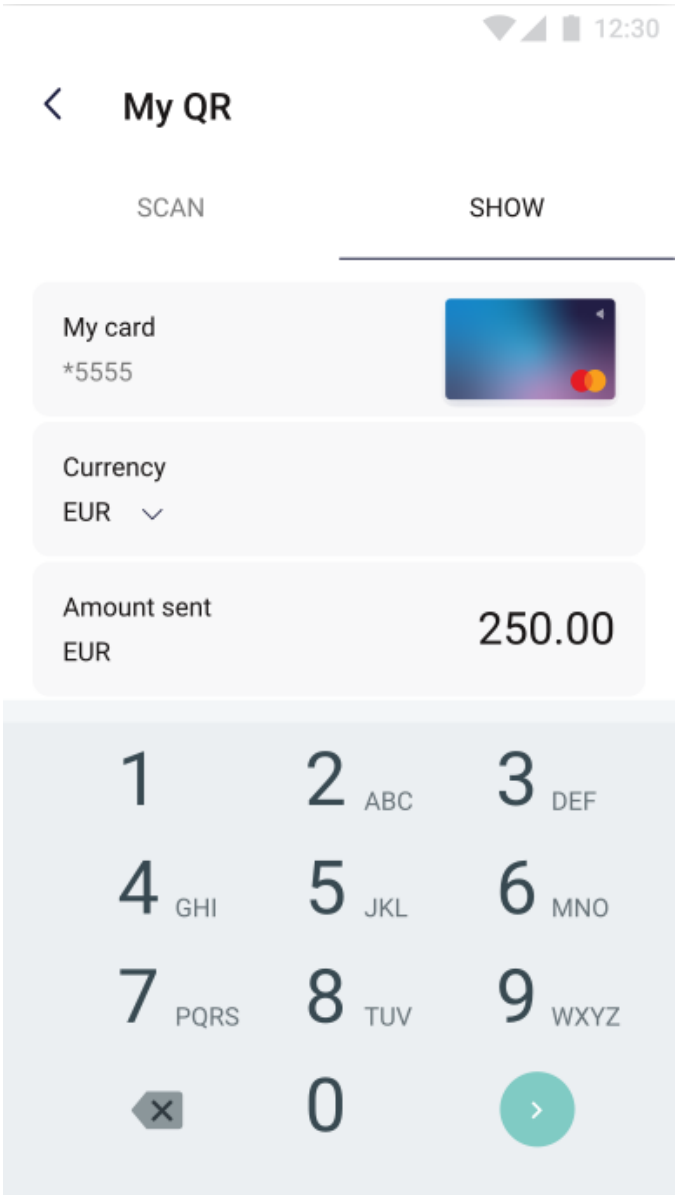
The scan QR option allows user to generate a QR code. With the generated code, the user can have his account topped up by showing the code to another person.

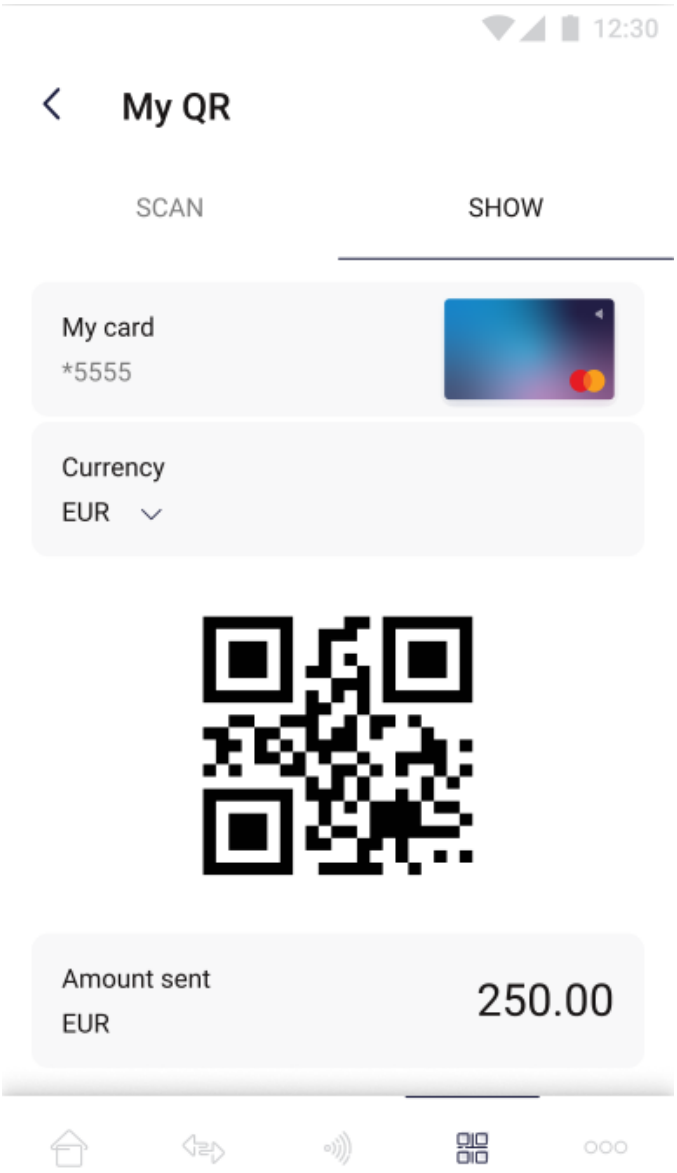
The pictures below show “generate QR code” view:





On the screen above user chooses which of his cards will be debited, and in what currency the transfer is to be made. If the above choices have been made, the user has to enter the amount he wants to receive.





On the screen above the user enters the amount he wants to receive and then confirms the selection with the green button in the lower right corner of the screen. After performing all the required actions, a QR code and information about the ordered transaction (card, currency and amount) are generated on the screen.

Revision #20
Created 20 June 2022 05:08:50
Updated 21 February 2023 12:11:34